195 PTAS.
(IVA Incluido)



104

CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen IX-Fascículo 104

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Aribau, 185, 1.°, 08021 Barcelona
Tel. (93) 209 80 22 - Télex: 93392 EPPA

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 120 fascículos de aparición semanal, encuadernables en diez volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-181-X (tomo 9)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 148601
Impreso en España-Printed in Spain-Enero 1986

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (120 fascículos más las tapas, guardas y transferibles para la confección de los 10 volúmenes) son las siguientes:

a) Un pago único anticipado de 27 105 ptas. o bien 10 pagos trimestrales anticipados y consecutivos de 2 711 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S. A. (Aribau, 185, 1.°, 08021 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

No se efectúan envíos contra reembolso.





Página impresa

Ya es posible diseñar una página como ésta en la pantalla de un micro. Estudiemos el proceso

La invención de la imprenta fue un acontecimiento de suma importancia, porque gracias a ella la palabra escrita se puso al alcance de una cantidad mucho mayor de personas. Antes de ella, por supuesto, los únicos libros disponibles se producían artesanalmente, ya sea escritos a mano, o bien compuestos con bloques de madera en los que se había grabado cada una de las letras. Esto significaba que existían muy pocas copias de cualquier volumen y que el costo de su producción era muy elevado. El procedimiento de impresión con caracteres móviles inventado hacia 1440 por el impresor alemán Johannes Gutenberg redujo drásticamente los costos de producción de cada unidad, de modo que los ejemplares de cada obra fueron a partir de entonces mucho más asequibles a la población.

No es razonable equiparar el significado de los últimos avances en el campo editorial con la invención de la imprenta, pero no obstante existen algunas analogías. Básicamente, los nuevos métodos de producción basados en la tecnología del ordenador pueden producir material de calidad similar al producido mediante los métodos de impresión tradicionales, aunque con un costo muy reducido. Pero si bien la iniciativa de Gutenberg puso los libros a disposición de un público más amplio, es probable que el proceso de producción informatizada tenga un efecto diferente, aunque de ningún modo menos importante.

Para las empresas editoras, la reducción de los costos de producción significaba tener que vender menos ejemplares para recuperar los gastos de su producción. Esto podría llevar a un aumento de la cantidad de publicaciones especializadas dirigidas a un público pequeño, antes inviables desde el punto de vista económico.

En el capítulo anterior nos referimos al proceso general que supone la producción de una revista y vimos algunos de los cambios que están teniendo lugar actualmente en ese campo. Llevando la idea de un sistema de producción modernizado un paso más hacia adelante, desviaremos nuestra atención ahora a un único microordenador independiente que se puede utilizar como procesador de textos, diseñador de maquetas y controlador de una impresora de gran calidad.

El Apple Macintosh, equipado con 512 Kbytes de memoria, ha dado lugar al desarrollo de varios paquetes sofisticados de software para usar con él, teniendo en mente precisamente esta finalidad. Entre éstos se incluye un paquete denominado *PageMaker*; si se le añade a este paquete una impresora de primera categoría, se crean las condiciones para que un pequeño grupo de personas produzca

Tony Sleep

Imagen digitalizada
Los digitalizadores permiten que el usuario almacene y luego reproduzca imágenes a utilizar en el diseño de páginas. Aunque antes su elevado precio los hacía inalcanzables, en la actualidad ya se venden a un precio razonable. La imagen digitalizada de la fotografía que vemos aquí se produjo aplicando el sistema Thunderscan

publicaciones de una calidad casi tipográfica y significativamente más rápida y económicamente que por los métodos tradicionales.

Cómo trabaja el «PageMaker»

El Macintosh, con su enfoque de ventanas, iconos y ratón, constituye un entorno ideal para aplicaciones como *PageMaker*. Cuando se carga el *PageMaker* se presentan los familiares menús y ventanas de pantalla, y la pantalla del Mac queda preparada como un escritorio completo con su caja de herramientas para que trabaje el maquetista.

Normalmente el maquetista comienza a trabajar con una hoja pautada que es la página estándar de una publicación, e incluye líneas de posición que sirven de ayuda para colocar las columnas de texto, los titulares, la numeración de las páginas, etc. Luego se emplean copias del papel pautado para componer cada página de la revista o periódico, cuidando que la misma tenga una apariencia uniforme.

Tradicionalmente, cuando se lanza una publicación, el maquetista traza a mano la hoja pautada, que después se adopta como estándar hasta que se decide un nuevo estilo.

Este tradicional método de diseño se reproduce en el *PageMaker*, que permite definir *páginas maestras*. Para éstas se selecciona el tamaño de la página (p. ej., A4) y la orientación (vertical o apaisada), se posicionan las guías de columnas, se define la anchura de los márgenes y se marcan las posiciones para los números de página, logotipos y titulares. Una vez preparada la página maestra, en el futuro podrá recuperarse en cualquier momento para que se constituya en la hoja pautada del maquetista.

En este punto se pueden maquetar las páginas individuales de la publicación. Un maquetista utilizará un escalpelo para cortar y posicionar texto e imágenes en la hoja pautada, mientras que el *PageMaker* permite cargar texto y gráficos desde disco. Y, en virtud de su compatibilidad con otros paquetes para el Mac, se podría escribir un artículo con el procesador de textos *MacWrite*, cargarlo directamente en *PageMaker* y posicionarlo en el tablero. Los gráficos diseñados con el *MacPaint* y el *MacDraw*, así como numerosas ilustraciones de esta publicación, también se pueden cargar mediante el *PageMaker* y posicionar en la página.

Tras haber cargado los diversos componentes que compondrán la página, se pueden utilizar los diversos dispositivos que contiene la caja de herramientas de la esquina de la pantalla.

Estas herramientas, por ejemplo, permiten añadir o corregir textos en pantalla seleccionando el icono A de la caja de herramientas o, seleccionando el icono «herramienta de siega», reducir, ampliar o recortar los gráficos para que quepan en el espacio disponible.

Utilizando el ratón se pueden recoger texto y gráficos y desplazarlos a través de la hoja pautada a voluntad, e incluso «dejar» temporalmente sobre la mesa escritorio mientras se lleva a cabo cualquier otra tarea.

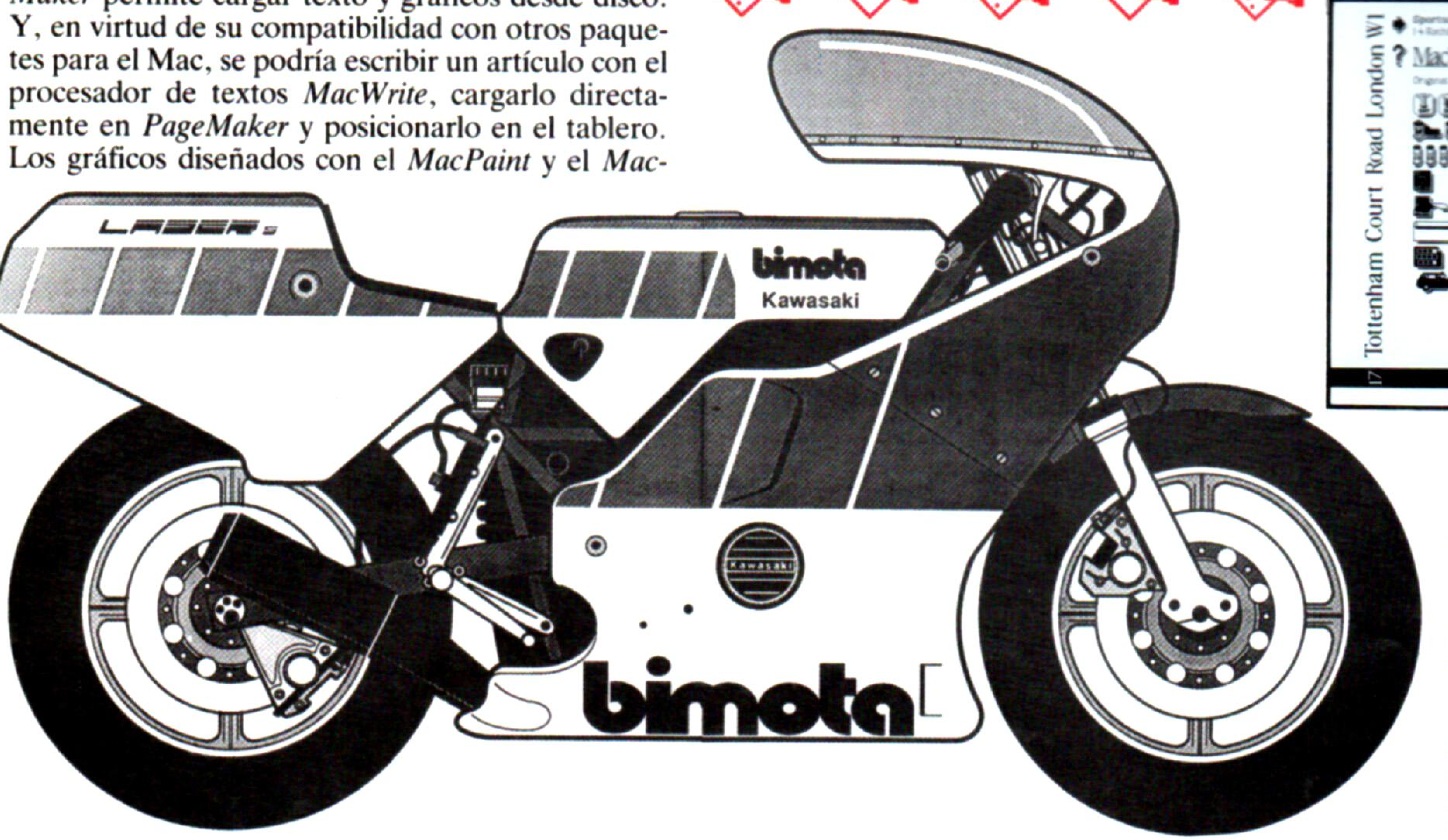
La caja de herramientas del *PageMaker* también incluye facilidades que permiten añadir toques especiales a la página, como dibujar ribetes alrededor de secciones específicas.

Incluye iconos que controlan el trazado de líneas, círculos, elipses y recuadros con o sin esquinas redondeadas.

También se puede seleccionar el espesor de las líneas extrayéndolo del menú adecuado. Estas formas, una vez dibujadas, se pueden rellenar en blanco o negro, varias tonalidades de gris o con formatos. A pesar de ser una herramienta increíblemente útil para los maquetistas profesionales, el *PageMaker* es suficientemente simple como para que el maquetista que no tiene experiencia lo utilice con todo éxito.

El costo de un Macintosh, una LaserWriter y el software para procesar la página necesarios para crear un sistema de producción con un micro es relativamente elevado, pero la introducción de máquinas más económicas basadas en el WIMP, como el Atari 520ST (véase p. 2029), que incorpora el

Iconos de facturas
Esta factura se diseñó y se imprimió usando una copia generada directamente por un Apple Macintosh. El diseño de iconos suele ser una configuración importante de publicaciones tales como *Mi Computer*, ya que facilitan la identificación de las diferentes secciones. De modo similar, los logos y símbolos de productos de las empresas con frecuencia se utilizan en facturas y membretes



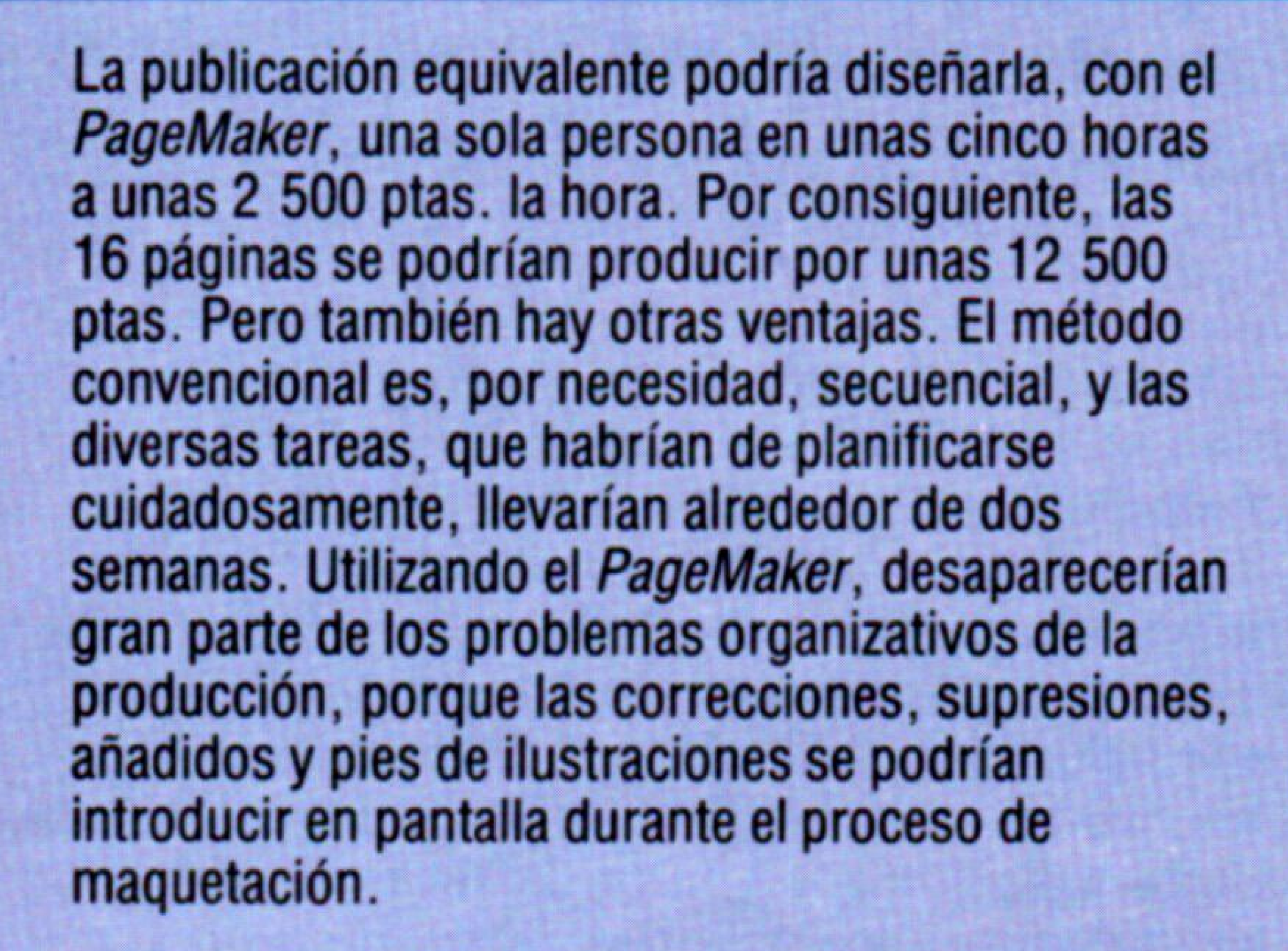
Pixels compatibles
Los archivos creados mediante programas como el *MacDraw* y el *MacPaint* se pueden almacenar en disco y después cargar en el *PageMaker* y manipularlos antes de su impresión. Tanto el *MacDraw* como el *MacPaint* ofrecen al maquetista poderosas facilidades, que se pueden utilizar para crear ilustraciones como ésta

Ian McKinnell

entorno GEM (administrador del entorno gráfico), probablemente dará como resultado importantes reducciones del precio de tales sistemas. Aun con los precios actuales, y teniendo en cuenta los costos de trabajo que se ahorran al producir una publicación utilizando sistemas informatizados de diseño de páginas, supone un significativo recorte de los costos de producción convencionales.

Rebajando los costos

¿Cuánto tiempo y dinero se podría ahorrar utilizando un sistema de procesamiento electrónico de páginas como el del Macintosh con la *LaserWriter* y el *PageMaker*? Imaginemos que se está aplicando este sistema para producir el boletín informativo de una empresa, que consta de 16 páginas de calidad profesional. Suponiendo que la publicación la produzca un pequeño equipo de maquetistas y redactores contratado por la empresa, y que todo el trabajo, exceptuando el tipográfico, se realice «en casa», el siguiente detalle representa los costos típicos de producción:



La publicación equivalente podría diseñarla, con el *PageMaker*, una sola persona en unas cinco horas a unas 2 500 ptas. la hora. Por consiguiente, las 16 páginas se podrían producir por unas 12 500 ptas. Pero también hay otras ventajas. El método convencional es, por necesidad, secuencial, y las diversas tareas, que habrían de planificarse cuidadosamente, llevarían alrededor de dos semanas. Utilizando el *PageMaker*, desaparecerían gran parte de los problemas organizativos de la producción, porque las correcciones, supresiones, añadidos y pies de ilustraciones se podrían introducir en pantalla durante el proceso de maquetación.

Tarea	**Precio**	**Costo ptas**
Preparación tipográfica	5 horas a 800 ptas/h	4 000
Composición y correcciones		20 000
Coordinación	5 horas a 800 ptas/h	4 000
Corrección de galeradas	8 horas a 400 ptas/h	3 200
Maqueta y diseño	6 horas a 800 ptas/h	4 800
Maqueta final	16 horas a 800 ptas/h	12 800
	Costos totales	**48 800**



Fijando el paso

Las modernas técnicas de impresión utilizan placas fotográficas para transferir una imagen al papel. Por lo tanto, las páginas preparadas para prensa han de estar «listas para la cámara» y, por cuanto concierne al texto, ello supone la producción de una imagen perfecta del texto a imprimir, al contrario que en el tipo en relieve que puede producir una linotipia de caracteres fundidos.

Las máquinas de fotocomposición hacen exactamente esto, aceptando texto como entrada desde un teclado o un disco y produciéndolo en forma de «galeradas», largas tiras de papel fotográfico en el cual se ha impreso el texto perfectamente (o así cabría esperar), listo para confeccionar la maqueta. Las primeras máquinas de fotocomposición utilizaban una plantilla giratoria en forma de disco que retenía una copia de todos los caracteres de un determinado tipo. El papel fotosensible pasaba sobre una cara del disco y, cuando la letra deseada giraba colocándose en posición, se encendía una luz, transfiriendo el carácter al papel. Aunque primitivos, estos sistemas alcanzaron considerables niveles de sofisticación y podían generar texto con moderada rapidez.

En la fotografía vemos un moderno sistema de fotocomposición basado en láser y producido por una Linotype-Paul, ejemplo típico de los sistemas controlados por ordenador, capaces de generar textos a 368 000 caracteres por hora.

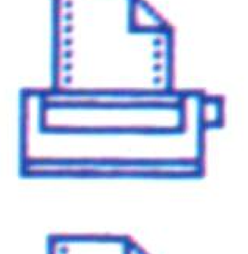
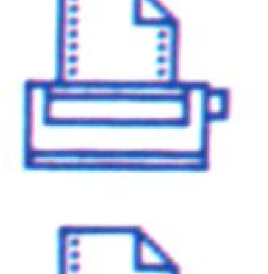
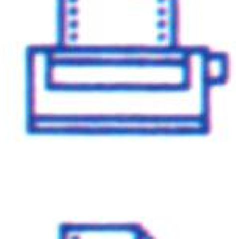
El sistema ilustrado posee tres componentes fundamentales. A la izquierda se halla la Linotronic 300, una máquina tipográfica láser de formato ancho y gran calidad, con una resolución de hasta un millón de pixels por cm^2. Los tipos se pueden ampliar o condensar, girar e invertir como se desee, y puede haber hasta 32 tipos de letra por línea. El terminal del centro permite entrar texto de forma manual, desde disco flexible o incluso a través de un modem. Esto último es útil para periodistas, quienes con frecuencia necesitan enviar texto al sistema mientras cubren una noticia en el mismo lugar en que se produce.

La unidad de la derecha, una Linotype-Paul Typeview 300, visualiza el texto en el estilo, tamaño y posición que ocupará en el montaje final. Estos sistemas no sólo permiten producir copias listas para la cámara, sino que también se pueden enlazar con sistemas en red mayores, a los que pueden acceder todo tipo de editoriales. Linotype ofrece un sistema capaz de manipular hasta 40 terminales interactivos a la vez, junto con un inmenso almacenamiento en línea y acceso a periféricos compartido. Por tanto, un sistema único, utilizado por un periódico, podría aceptar entradas de los reporteros desde su puesto de trabajo en el periódico o desde fuera, recuperar material gráfico almacenado, pasarlo a los redactores, combinarlo con trabajos artísticos y luego presentar el material al departamento de maquetación antes de producir una placa para su procesamiento e impresión














Máquina versátil

En este penúltimo capítulo analizamos los accesorios que podrían aumentar la versatilidad del tester

Tal como está, el tester digital puede medir tensiones de entre 0,0000 V y 1,9999 V, motivo por el cual decimos que posee una sensibilidad básica de 2 V. Atenuando la señal a medir, teóricamente sería posible medir tensiones de hasta 19 999 V (20 kV). Estas tensiones tan altas pueden ser sumamente peligrosas, de modo que hemos limitado la escala máxima a 199,99 V.

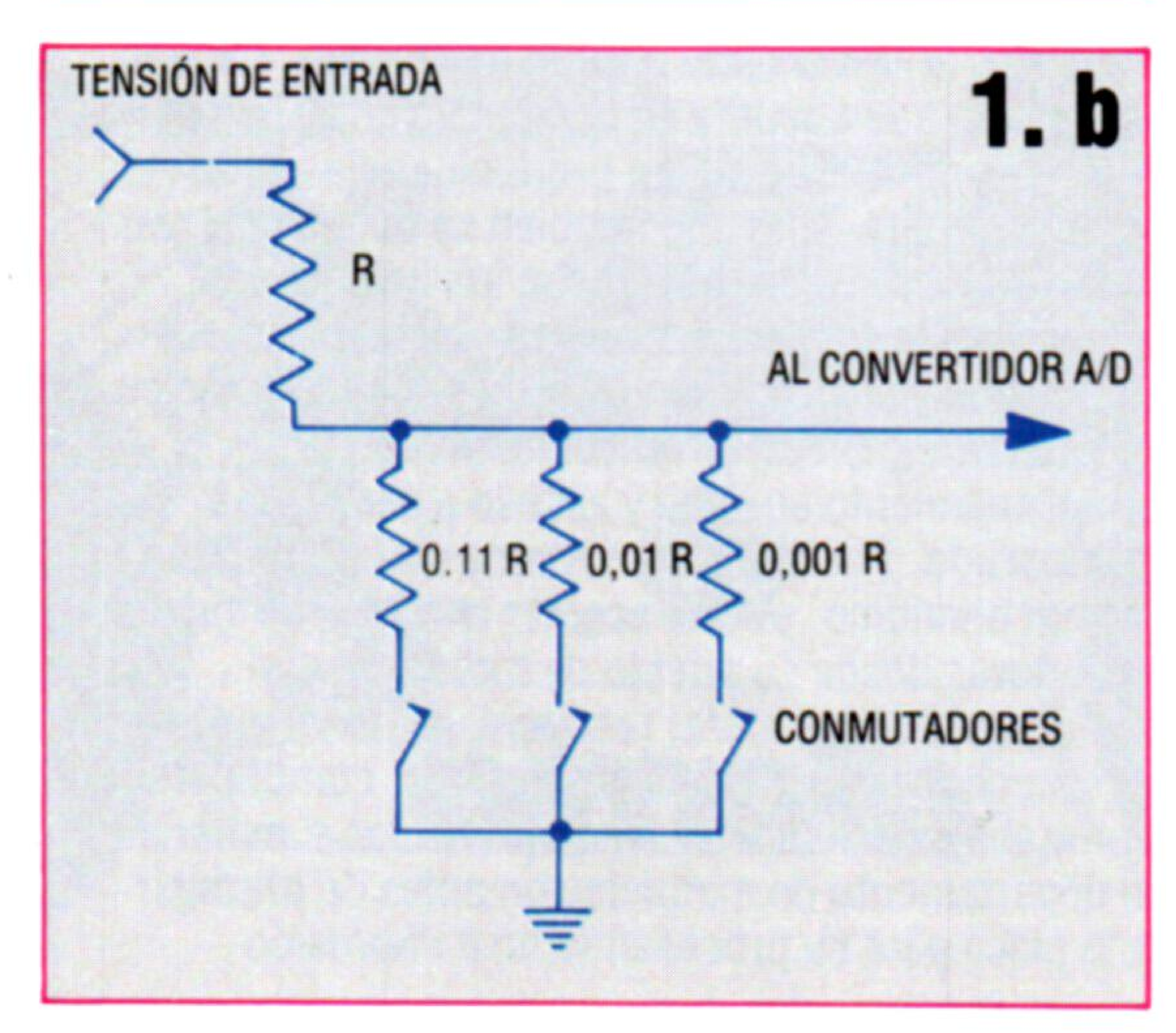
Quizá haya notado, a partir del diagrama del capítulo anterior, que había tres cables sin conectar, procedentes de tres de los LED. Éstos se conectan a los puntos decimales que aparecen a la derecha del dígito de los LED y se utilizan para encender el punto decimal cuando así se requiere. Un conmutador de tres posiciones y de un polo, «acoplado» al conmutador de atenuador de entrada, asegura que esté encendido el punto decimal adecuado para la sensibilidad seleccionada. Consideremos primero el circuito del atenuador de entrada.

Básicamente hay dos formas de hacer un atenuador de entrada. Ambas implican el empleo de un circuito divisor de potencial que distribuye la tensión de entrada en varias resistencias, derivando una fracción de la tensión de la entrada para la medición. En el diagrama 1 se muestran estos dos tipos de atenuadores. El primer tipo ofrece la ventaja de la simplicidad, además de ser fácilmente adaptable a la medición de corriente y resistencia. Esto no es así en el caso del segundo atenuador. En un circuito de escala automática (en el cual la atenuación de entrada se conmuta automáticamente), el segundo tipo ofrece la ventaja de poder utilizar conmutadores analógicos de estado, mientras que el primero sólo puede utilizar conmutadores mecánicos. En nuestro caso utilizaremos el segundo

Diseños de atenuadores
Para aumentar la escala de nuestro voltímetro básico necesitamos diseñar un atenuador de entrada que reduzca la escala de tensión a un nivel aceptable para el circuito de conversión A/D principal. Ambos utilizan varias resistencias que se pueden conmutar en el circuito para derivar una proporción del potencial de entrada. Nosotros utilizaremos el primero de los dos diseños que vemos aquí, porque el segundo diseño tiene el inconveniente de requerir un circuito de protección de entrada especial (si bien puede incorporar conmutadores analógicos de estado sólido)

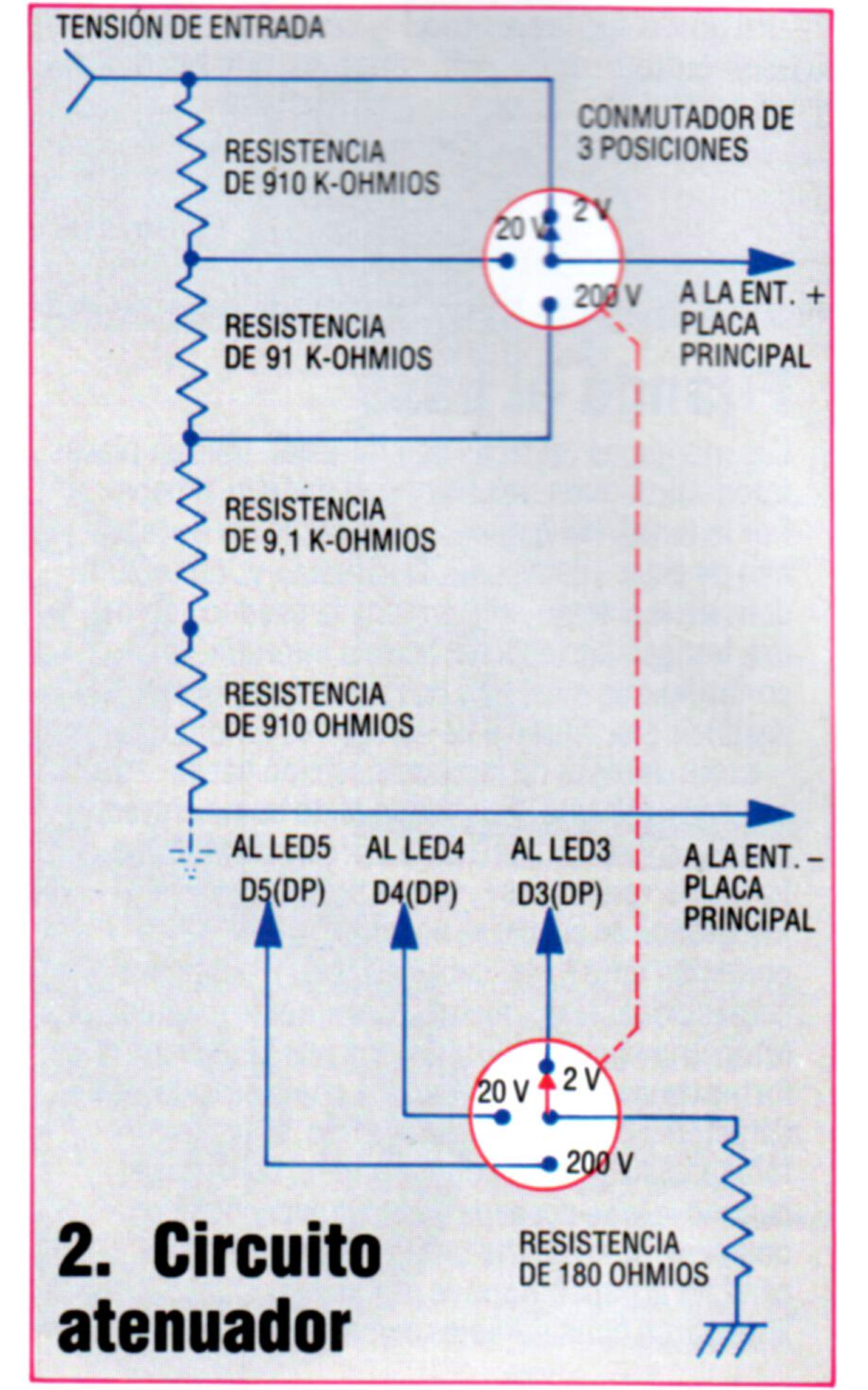
TENSIÓN DE ENTRADA
CONM. 2 V
AL CONVERTIDOR A/D
R
CONM. 20 V
R/10
CONM. 200 V
R/100
R/1000

1a. Diseños de atenuadores

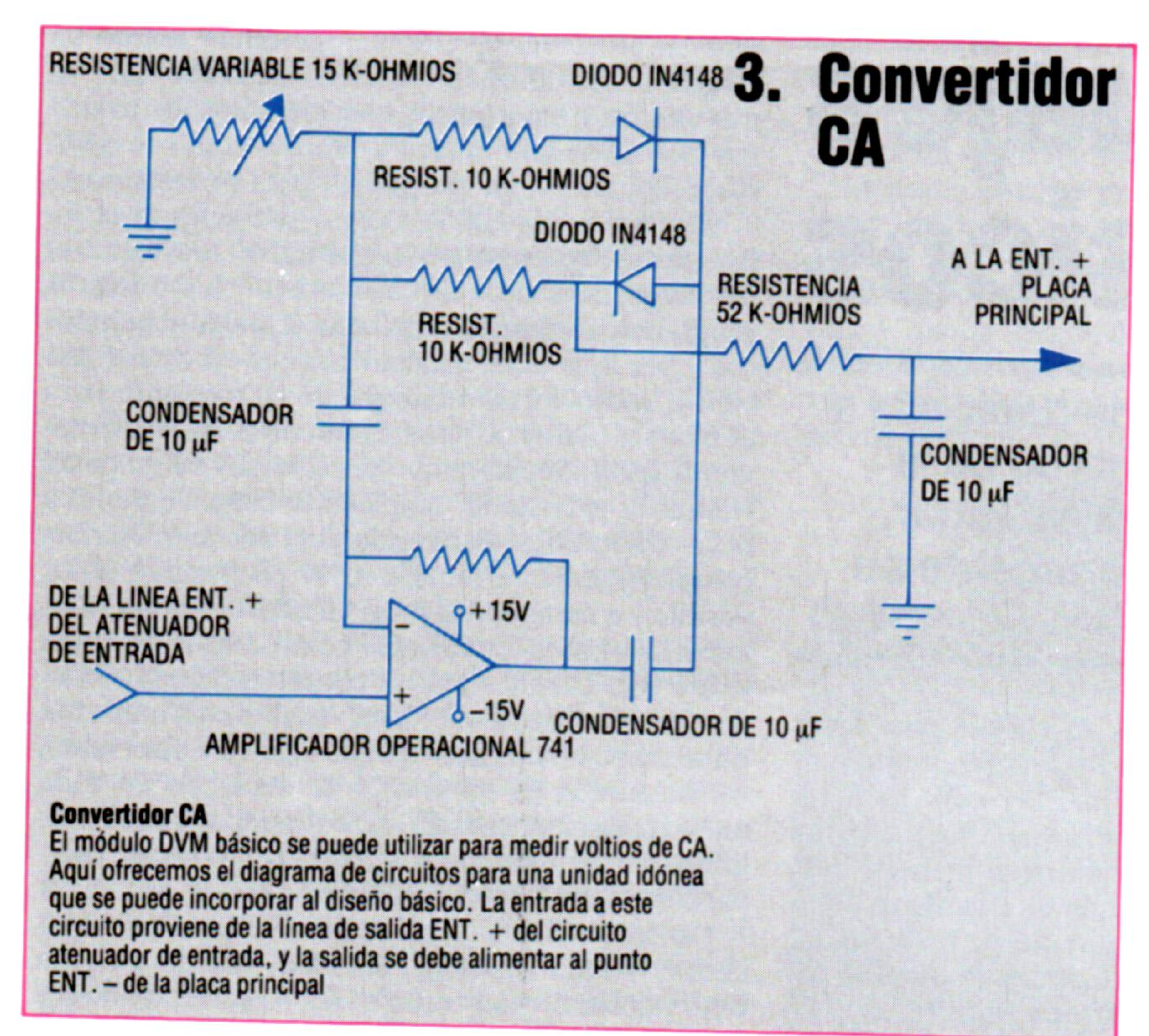


1. b

Circuito atenuador
Para construir el atenuador de entrada emplee un pequeño trozo de placa matriz de 0,1 pulgadas. Las resistencias han de ser del 1 % de tolerancia sobre los valores nominales. Monte las resistencias a lo largo de una línea de la placa y derive todos los extremos de las resistencias de 910 K-ohmios y 91 K-ohmios a un conmutador de tres circuitos y tres polos. Las salidas negativa y positiva se deben conectar a la placa principal mediante unos trozos cortos de hilo aislado. Los tres terminales restantes del conmutador de dos circuitos se deben conectar a las patillas del punto decimal de los LED 3, 4 y 5 del visualizador. Estas conexiones permiten reposicionar automáticamente el punto decimal de la visualización cuando se selecciona una nueva escala



2. Circuito atenuador



Convertidor CA
El módulo DVM básico se puede utilizar para medir voltios de CA. Aquí ofrecemos el diagrama de circuitos para una unidad idónea que se puede incorporar al diseño básico. La entrada a este circuito proviene de la línea de salida ENT. + del circuito atenuador de entrada, y la salida se debe alimentar al punto ENT. – de la placa principal

tipo, dada la sencillez con que se puede adaptar para realizar mediciones.

El verdadero circuito (diagrama 2), junto con la conmutación del punto decimal, se puede construir fácilmente en un pequeño trozo de placa matriz, sustituyendo y conectando los componentes especificados en la lista de componentes por los del diagrama de circuitos. Las resistencias especificadas son de tipo de película metálica, de un 1 % de tolerancia.

El trazado no es crítico, pero es aconsejable que todos los trozos de hilo sean lo más cortos posible, puesto que la impedancia de entrada del chip A/D es muy alta y los hilos largos actúan como una antena para señales espurias.

Este circuito atenuador es de diseño simple y utiliza valores de resistencias fáciles de conseguir. Sin embargo, posee la desventaja de presentar una carga relativamente baja para el circuito que se esté probando (poco más de 1 M-ohmio); pero ésta no es una limitación grave: si se está midiendo una señal de 5 V, representa una corriente de carga de alrededor de 5 μA.

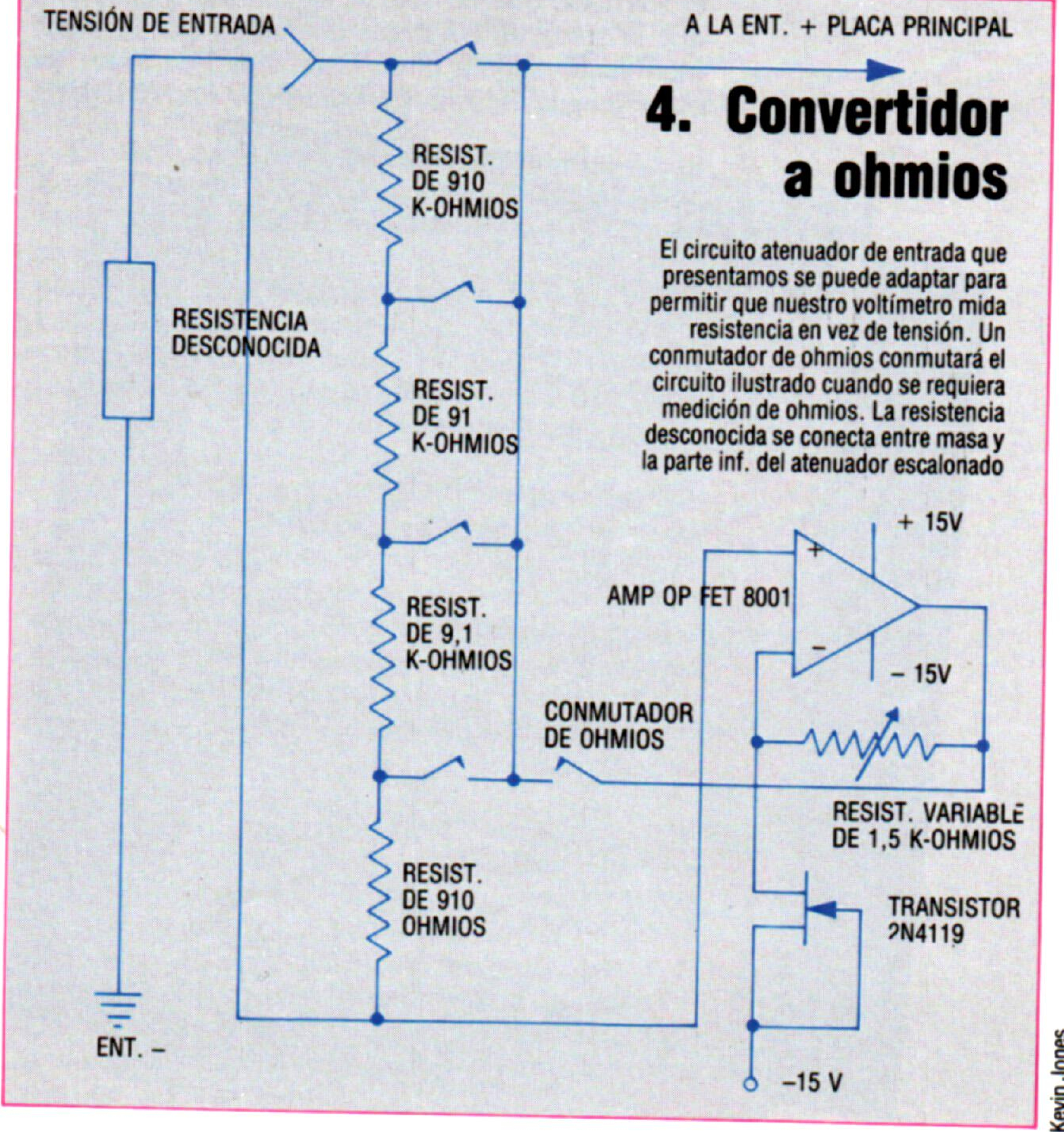
El circuito que hemos presentado puede medir voltios de CC en tres escalas: 2 V, 20 V y 200 V. Modificaciones o adiciones relativamente simples al circuito básico permitirán medir muchas otras unidades, incluyendo ohmios, tensiones CC y temperatura.

Ofrecemos aquí algunas posibles adiciones al circuito básico, pero solamente en forma de diagrama de circuitos.

Para medir voltios CA usando un voltímetro CC (como nuestro DVM), usted sólo necesita rectificar la señal CA a CC y medirla. Lamentablemente, no servirá un rectificador de diodo simple; lo que se requiere es uno de los llamados rectificadores de precisión. El diagrama 3 muestra un posible circuito. Se basa en el amplificador operativo integrado 741, muy conocido y de precio muy bajo. En este caso, la única desventaja es la necesidad de una fuente de alimentación de ±15 V, pero la misma se podría derivar de la salida CC de 12 V del transformador de la red; las exigencias de corriente del 741 son muy bajas.

Medir ohmios en un voltímetro analógico convencional es fácil, dado que la caída de tensión en una resistencia que se está midiendo es proporcional al valor de esta última. Sin embargo, eso no es tan fácil con un voltímetro digital y, en consecuencia, se requiere un circuito convertidor a ohmios. El convertidor de ohmios (ver diagrama 4) funciona aplicándole una tensión constante al atenuador de entrada escalonado, que genera una corriente constante en la resistencia que se está midiendo. Puesto que una corriente constante a través de una resistencia desconocida genera en ella una tensión proporcional a su valor, el DVM podrá leer directamente esta caída de tensión. Nuevamente se requerirá un amplificador operacional, y habrá de ser uno que tenga un tipo de impedancia de entrada muy elevada.

Mediante la aplicación de la salida del convertidor de ohmios al atenuador escalonado se pueden medir cuatro escalas de ohmios: 2 K-ohmios, 20 K-ohmios, 200 K-ohmios y 2 M-ohmios. (Por esta razón se utilizaron dos resistencias separadas en la parte inferior del divisor escalonado, en lugar de las resistencias individuales de 10 K-ohmios que hubieran sido adecuadas si sólo se hubiesen requerido tres escalas de tensión.) Como es natural, se requerirá un conmutador de atenuación de cuatro circuitos en lugar del interruptor de tres circuitos especificado.



El circuito atenuador de entrada que presentamos se puede adaptar para permitir que nuestro voltímetro mida resistencia en vez de tensión. Un conmutador de ohmios conmutará el circuito ilustrado cuando se requiera medición de ohmios. La resistencia desconocida se conecta entre masa y la parte inf. del atenuador escalonado

Lenguajes clásicos

Iniciamos una serie dedicada a los lenguajes pioneros de la informática. En primer lugar, revisaremos someramente su historia y desarrollo

Para la mayoría de nosotros, la idea de programar un ordenador constituye un hecho corriente. Aunque todavía no podamos hacerlo muy bien, al menos sabemos de qué se trata. Las cosas eran muy diferentes, sin embargo, a finales de los años cuarenta y principios de los cincuenta, cuando se desarrolló por primera vez el concepto de un lenguaje de programación. Este primer período produjo algunos lenguajes que aún hoy se utilizan.

Esta serie de artículos tiene por objeto considerar la evolución de estos lenguajes hasta alcanzar su forma actual, así como rastrear algunas de las influencias que tuvieron en lenguajes aparecidos posteriormente y descubrir su aportación a los microordenadores modernos. Hablaremos con mayor profundidad de dos de estos lenguajes, el COBOL y el FORTRAN, que no sólo se siguen utilizando, sino que la suma de las líneas de código que se están escribiendo supera la de la totalidad de todos los otros lenguajes juntos. Asimismo, nos detendremos en el ALGOL, que si bien continúa siendo un lenguaje importante, ha sido superado por el PASCAL. En primer lugar, consideremos los primeros días de la informática y veamos cómo se desarrolló la idea de un «lenguaje» para ordenadores.

Si pasamos por alto las etapas más tempranas, en las cuales «programar» significaba recablear el hardware para cada tarea diferente, los primeros programas auténticos eran completamente numéricos y por lo general estaban escritos en octal (base ocho), como forma taquigráfica conveniente para el binario. Estos programas se entraban laboriosamente mediante interruptores situados en un panel frontal o, más tarde, empleando cinta de papel o fichas perforadas. Cada programa nuevo se escribía completamente de la nada en el nivel más inferior posible y a menudo los programadores se encontraban a sí mismos escribiendo las mismas rutinas una y otra vez. De allí surgió el concepto de una biblioteca de subrutinas, si bien el vocablo *subrutina* aún no se empleaba. Estas «bibliotecas» se conservaban en cuadernos de apuntes y se copiaban en cada nuevo programa cuando así se requería, y cada programador contaba con sus propias bibliotecas, compartiéndolas sólo ocasionalmente.

La construcción del sistema EDSAC en Manchester (Gran Bretaña), en 1951, supuso un gran paso adelante, y el trabajo de Wheeler, Wilkes y Gill se plasmó en un conjunto coherente de subrutinas generales para la máquina. Ahora que todos utilizaban las mismas subrutinas, los programas fueron más fáciles de escribir y comenzaron a traslucir semejanzas estructurales con sus equivalentes modernos. Unos bloques de código realizaban la tarea determinada con llamadas a las subrutinas para llevar a cabo las funciones estándares tales como entrada, salida y cálculos numéricos, que eran comunes a la mayoría de los programas.

Cuando aumentó la capacidad de memoria de las máquinas, se hizo posible almacenar las bibliotecas de subrutinas internamente, o al menos en línea, y a partir de allí sólo hubo un corto paso hacia el desarrollo de un código que permitiera al programador especificar las subrutinas requeridas utilizando caracteres alfabéticos y una notación matemática. Un programa observaría cada línea de este código, determinaría las subrutinas requeridas, las llamaría y luego regresaría a la siguiente línea. Ésta es la forma en que trabaja un intérprete de BASIC. Un ejemplo de esto fue el *Short code* (Código corto) que produjo John Maunchly en 1952 en el ordenador BINAC y luego en el UNIVAC.

Otro refinamiento de esta primera técnica tomaba cada subrutina cuando así se requería, pero en vez de ejecutarla directamente, primero la copiaba en un dispositivo de salida, acabando, por tanto, con un programa completo ejecutable. Fue durante esta etapa cuando se desarrollaron los primeros compiladores, con el término *compilar* aludiendo a la forma en que se conformaba el programa a partir de una cantidad de componentes dispuestos por un orden lógico, de la misma forma en que se compilaría un conjunto de ensayos.

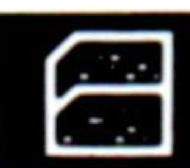
Fechas clave

Cronología simplificada del desarrollo de los primeros lenguajes de programación:

1951	Se completa en Manchester el EDSAC, primer ordenador digital con programas almacenados
1952	*Short code* de John Maunchly
1953	*Speedcoding* de Laning y Zierly
1954	Compilador A2. Primeras especificaciones para el FORTRAN
1955	Primeros compiladores auténticos para FORTRAN y FLOW-MATIC
1956	Lanzamiento generalizado de compiladores FORTRAN y FLOW-MATIC. Especificaciones para el FORTRAN II
1957	Especificaciones y lanzamiento del primer compilador de ALGOL
1958	Lanzamiento del FORTRAN II
1959	Primeras especificaciones para el COBOL. Primer compilador de LISP
1960	Lanzamiento del primer compilador de COBOL. Lanzamiento de la versión revisada ALGOL 60

Los primeros compiladores

Uno de los primeros compiladores que tuvieron éxito fue el A-2, desarrollado en Remington Rand en 1955 por un equipo dirigido por Grace Hopper.



El mismo utilizaba el concepto de *tres direcciones*, en el que cada operación tenía un nombre mnemotécnico seguido por tres direcciones: dos para los datos fuente y una para el destino. En algunos sentidos, era muy similar a un ensamblador, con la excepción de que las instrucciones no cabían en la arquitectura de ninguna máquina específica. Posteriormente este lenguaje se convirtió en ARITH-MATIC, al cual siguió un lenguaje similar de Remington Rand: el AT3, o MATH-MATIC.

A fines de 1951, numerosas personas habían comprendido que, desde el punto de vista del usuario externo, de hecho los ordenadores estaban ejecutando programas que se habían escrito utilizando un código distinto al código máquina nativo. El hecho de que el ordenador realizara primero una tarea de traducción era irrelevante.

En este caso, uno bien podría diseñar su propio lenguaje nuevo con el objeto de hacer más sencilla la tarea de escribir al programador, en lugar de facilitar la tarea de traducción, especialmente porque el hardware se estaba haciendo más rápido y grande y los programadores no podían seguirle el paso.

El siguiente problema que hubo que abordar luego fue que no había forma de estandarización, porque cada máquina utilizaba su propio lenguaje relacionado con un pequeño conjunto particular de problemas. En consecuencia, el paso siguiente fue el desarrollo de un lenguaje independiente de todas las especificaciones de cualquier hardware dado, para abordar una clase más amplia de problemas. Este enfoque se inició en 1954 y condujo al FORTRAN (sistema de IBM de traducción de fórmulas [FOR*mula* TRANS*lation*] matemáticas), el primer auténtico lenguaje de programación.

El FORTRAN es un lenguaje de base matemática, muy adecuado para el trabajo que se realizaba entonces, mayormente numérico, y similar a muchos de los autocódigos de la época. Sin embargo, aún se basaba en gran medida en las arquitecturas de las máquinas disponibles. La comunidad empresarial se mostraba cada vez más interesada en utilizar los ordenadores para el procesamiento de datos a gran escala, pero necesitaban un lenguaje que se pareciera más al inglés comercial común.

No deja de ser divertido que en aquel entonces hubiera muchos programadores que pensaran que esto era imposible, ya que no existía forma imaginable de que el ordenador «comprendiera» palabras en lugar de números. Pronto esta falacia se hizo evidente, gracias (entre otros) a Grace Hopper, quien desarrolló un lenguaje denominado FLOW-MATIC, que podía ser leído y comprendido tanto por la dirección como por los programadores. Hacia 1956, este lenguaje se había convertido en el COBOL (CO*mmon* B*usiness* O*riented* L*anguage*). Estos lenguajes tenían especificaciones rígidas.

La teoría del lenguaje

Para entonces el número de ordenadores y de programadores había aumentado espectacularmente y la gente comenzaba a pensar más en el aspecto teórico de los lenguajes de programación, intentando hallar la forma más eficaz y elegante de expresar los algoritmos. Esto llevó, en 1958, al desarrollo del ALGOL (ALGO*rithmic* L*anguage*: lenguaje algorítmico). El ALGOL nunca alcanzó la amplia popularidad del FORTRAN o el COBOL; pero ocupa un importante lugar en el desarrollo de lenguajes. Fue el ALGOL el primero que incorporó el principio de diseño sólido de programa, que ha sido una consideración fundamental en todos los lenguajes posteriores.

Durante aquellos primeros días se desarrollaron otros lenguajes, muchos de los cuales todavía están en uso. Un de los mejores ejemplos es el LISP (LIS*t* P*rocessing language*: lenguaje de procesamiento de listas), que se desarrolló entre 1956 y 1958. No sólo se sigue utilizando en la actualidad, sino que tiene una importancia creciente en el campo de la inteligencia artificial. Los tres lenguajes, FORTRAN, COBOL y ALGOL, han representado, sin embargo, la principal tendencia de la programación durante las dos últimas décadas. Con el correr de los años han sido objeto de algunas revisiones para incorporar nuevas facilidades y reflejar las modernas tendencias del diseño de lenguajes. Las revisiones actuales son el FORTRAN 77 (especificación que entró en vigor en 1977), el COBOL 74 y el ALGOL 68.

En 1964, en el Dartmouth College (Estados Unidos) se introdujo una versión muy simplificada del FORTRAN (con algunas influencias del ALGOL), para simplificar mucho más la tarea de aprender a programar y para utilizar los nuevos sistemas multiusuario de tiempo compartido que estaban apareciendo. Esta versión se dio a conocer como BASIC (B*eginners* A*ll-purpose* S*ymbolic* I*nstruction* C*ode*: código de instrucciones simbólicas con fines generales destinado al principiante).

Cuando se estaba preparando el nuevo estándar 1968 para el ALGOL, Niklaus Wirth se mostró en desacuerdo con la forma en que el lenguaje se estaba volviendo más complejo, y se mostró favorable a un enfoque más simple y elegante. Por lo general se piensa que el ALGOL 68 es demasiado complejo para uso normal y su presencia es rara fuera de las más altas esferas académicas. Sin embargo, la versión de Wirth, muy simplificada y denominada PASCAL, ha obtenido una gran difusión.

El COBOL no ha dado lugar a derivados de la misma forma, dado que se ha utilizado casi exclusivamente en su propio campo. El enorme volumen de programas que hay escritos en COBOL ha significado que se haya utilizado como vehículo para numerosos desarrollos en el área de diseño de programas y sistemas; por ejemplo, generación de programas, diseño de programas estructurados y empleo de bases de datos.

Código corto

En el *Short code* (Código corto) de John Maunchly la conocida sentencia de asignación de BASIC:

```
10 A = B + C
```

se escribiría

```
10 S0 03 S1'07 S2
```

donde 10 es el número de línea y S0, S1 y S2 son símbolos que representan las «variables» A, B y C como palabras individuales en la memoria. 03 y 07 son los códigos de operación para asignación y suma, respectivamente. En el lenguaje A2. la misma sentencia aparecería como:

```
ADD B C A
```

Taquilla

Sistemas simples de reserva de localidades:

FORTRAN IV:

```
C       RESERVA DE LOCALIDADES PARA EL
        TEATRO.
C       PROGRAMA PARA ACEPTAR UN
        NUMERO DE ASIENTO, COMPROBAR SI
        YA ESTA OCUPADO.
C       RESERVADO SI ESTA LIBRE O DAR
        MENSAJE SI YA ESTA RESERVADO.
C
C       DECLARAR VARIABLES.
C
        INTEGER NUMASIENTO, ASIENTO (500)
C       SEÑALAR TODOS LOS ASIENTOS
C       DISPONIBLES.
C
        DO 100 I = 1,500
100     ASIENTO (I) = 0
C
C       LEER NUMERO DE ASIENTO Y
        COMPROBAR DISPONIBILIDAD
C
101     READ (1,10) NUMASIENTO
        IF (ASIENTO(NUMASIENTO).NE.0)GOTO
        102
C
C       EL ASIENTO ESTA DISPONIBLE.
C
        ASIENTO(NUMASIENTO) = 1
        WRITE(1,20)
        GOTO 103
C
C       EL ASIENTO ESTA RESERVADO
C
102     WRITE(1,30)
C
C       SIGUIENTE NUMERO DE ASIENTO
C
103     GOTO 100
C
C       SENTENCIAS FORMAT
C
10      FORMAT(I4)
20      FORMAT(1H,17HASIENTO ESTA LIBRE)
30      FORMAT(1H,19HASIENTO YA ESTA
        RESERVADO)
        END
```

ALGOL 60:

```
comment RESERVA DE LOCAL. PARA EL TEATRO.
PROGRAMA PARA ACEPTAR UN NUMERO DE
ASIENTO, COMPROBAR SI YA ESTA RESERVADO,
RESERVARLO SI ESTA LIBRE O DAR MENSAJE SI
YA ESTA RESERVADO
OBSERVE QUE EL ALGOL NO INCLUYE
SENTENCIAS DE ENTRADA/SALIDA:
  begin
    integer NUMASIENTO, I;
    integer array ASIENTO[1:500];
    comment MARCAR LOS ASIENTOS LIBRES
    for I:=1 step 1 until 500 do
      ASIENTO[I]:=0
    comment LEER NUMERO DE ASIENTO Y
    COMPROBAR DISPONIBILIDAD;
  NUEVO ASIENTO:((read NUMASIENTO));
    if ASIENTO[NUMASIENTO]=0 then
    begin
        ASIENTO[NUMASIENTO]:=1;
        ((print 'EL ASIENTO ESTA LIBRE'));
      end
      else
        ((print 'EL ASIENTO YA ESTA
        RESERVADO'));
      goto NUEVOASIENTO;
  end
```

COBOL 74 [sólo división DATOS y PROCEDIMIENTO]:

```
*RESERVA DE LOCALIDADES PARA EL TEATRO.
*PROGRAMA PARA ACEPTAR UN NUMERO DE
ASIENTO, COMPROBAR SI YA ESTA RESERVADO,
*RESERVARLO SI ESTA LIBRE O DAR MENSAJE SI
YA ESTA RESERVADO.
DIVISION DE DATOS.
SECCION DE ALMACENAMIENTO OPERATIVO.
01  DISPONIBILIDAD-ASIENTO.
    02 ASIENTO PIC 9 OCCURS 500 TIMES.
77  NUM-ASIENTO PIC 999.
77  CONTADOR-BUCLE PIC 999.
77  ASIENTO-DISPONIBLE PIC X(17)VALUE
    'EL ASIENTO ESTA LIBRE'.
77  ASIENTO-RESERVADO PIC X(19)VALUE
    'EL ASIENTO YA ESTA RESERVADO'.
DIVISION PROCEDIMIENTO.
PARRAFO PRINCIPAL.
*MARCAR TODOS LOS ASIENTOS DISPONIBLES
    REALIZAR PARRAFO-MARCAR-ASIENTO-
      DISPONIBLE VARIANDO EL CONTADOR DEL
      BUCLE DE 1 EN 1 UNTIL I>500.
*LEER UN NUMERO DE ASIENTO Y COMPROBAR
    DISPONIBILIDAD.
    REALIZAR PARRAFO-TOMAR-NUMERO-
    ASIENTO.
    STOP RUN.
PARRAFO-MARCAR-ASIENTO-DISPONIBLE.
  MOVE ZERO TO ASIENTO (CONTADOR-BUCLE).
PARRAFO-TOMAR-NUMERO-ASIENTO.
  IF ASIENTO (NUMERO-ASIENTO) IS EQUAL TO 0
    MOVE 1 TO ASIENTO(NUMERO-ASIENTO)
    DISPLAY ASIENTO LIBRE
  ELSE
    DISPLAY ASIENTO RESERVADO
  GOTO PARRAFO-TOMAR-NUMERO-ASIENTO.
```

Reserva de localidades
Ofrecemos aquí listados en tres lenguajes de alto nivel distintos (FORTRAN, ALGOL y COBOL) demostrando la estructura del programa y algunas configuraciones comparativas. Cada programa acepta como entrada un número de asiento, comprueba si el número no se ha entrado ya (en cuyo caso estaría «ocupado») y, si no es así, señala que ese asiento ya tiene dueño

BBC Hulton Picture Library

Función casera

El Pioneer PX-7 puede combinarse con un aparato reproductor de discos láser para crear un sistema propio de video interactivo

Ya vimos las nuevas disponibilidades que ofrecen los discos láser controlados por ordenador en los campos de la educación y del ocio. El ordenador personal PX-7 de Pioneer, en unión con el reproductor de discos láser LD-700 (que vemos aquí) o el LD-1100, más sofisticado, ofrece la realización de esas posibilidades a un precio al alcance del usuario personal. El PX-7 es un ordenador MSX sumamente desarrollado, mientras que el LD-700 es un reproductor de discos láser de precio asequible que utiliza discos láser de 12 pulgadas estilo Philips.

El PX-7 es muy poco corriente tratándose de un ordenador MSX, siendo el primero de ellos que posee un teclado separado. Esto permite ubicar la unidad CPU junto a un grabador de video, un reproductor de discos láser, un televisor o un sistema de alta fidelidad. En consecuencia, el diseño de la unidad CPU responde más al de un componente de *hi-fi* que a un ordenador personal, y el teclado se conecta mediante un generoso cable de metro y medio. En la parte delantera de la máquina hay un control de volumen, un conector para auriculares y un control de mezcla, que regula el equilibrio entre el sonido generado por el ordenador y el proveniente de una fuente externa como el disco láser. Un conmutador de audio-video aísla efectivamente al ordenador, permitiendo que las señales de video y audio externas pasen directamente a un *hi-fi* o a un televisor conectados a la unidad.

En el interior hay un ordenador MSX estándar de 32 K, que se puede utilizar con la gama de software MSX y los periféricos existentes. Sorprendentemente, a pesar de la sofisticada tecnología del disco láser, el PX-7 utiliza cassettes para almacenar sus programas e información. Para quienes deseen ahondar más en el potencial del micro, un segundo conector para cartuchos en la parte posterior hará lugar para unidades de disco y otros accesorios.

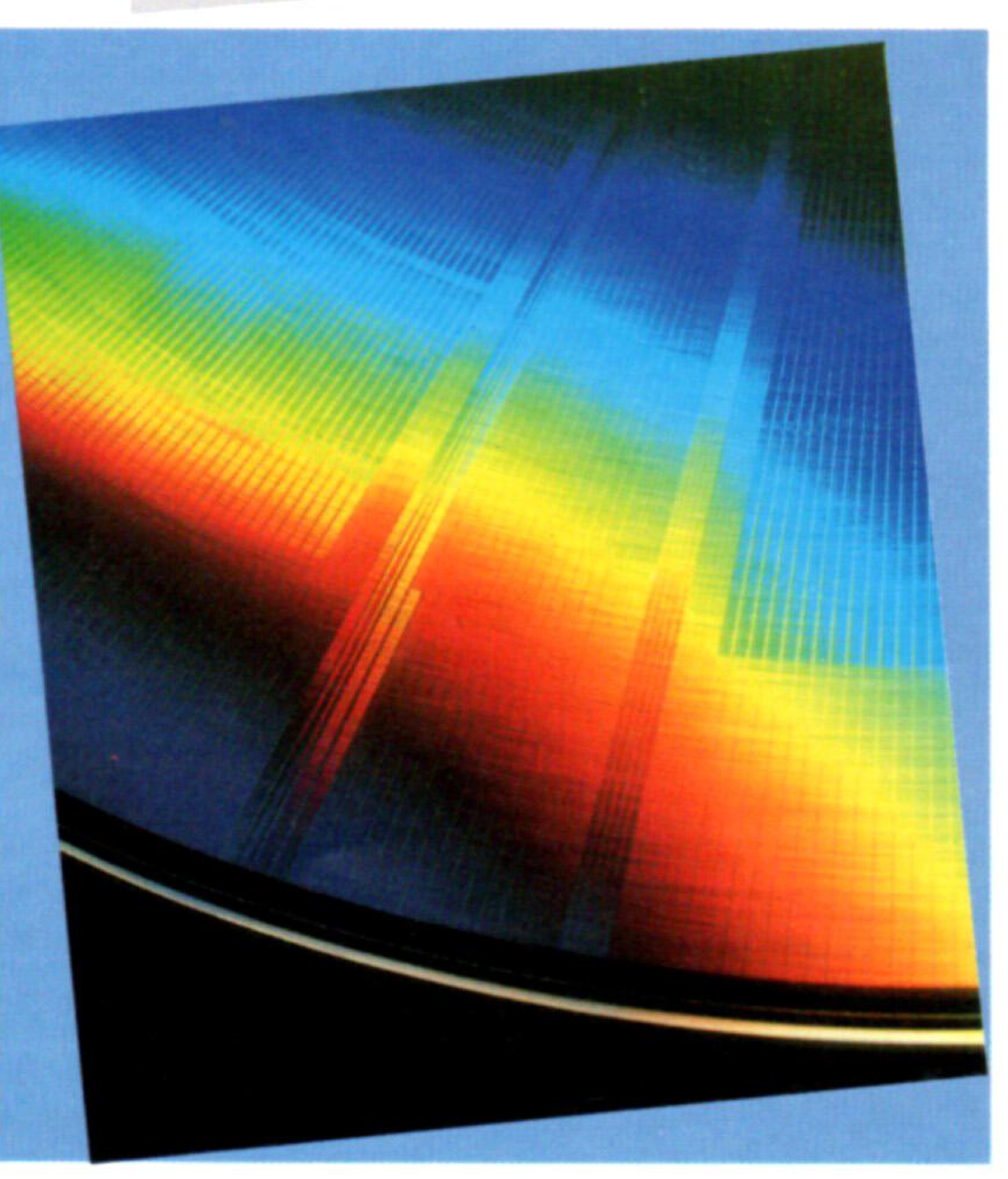
Hay una gama completa de interfaces de ordenador para televisión, pantalla compuesta o en color, cassette, palancas de mando *(joysticks)* gemelas, dos cartuchos y una impresora Centronics. Además, la unidad posee conexiones estéreo para un *hi-fi* y una interface para «control del sistema». Esta última es una interface para fines generales y se puede usar para conectar reproductores de discos láser, grabadores de videocassettes, etc. A medida que vayan apareciendo nuevos equipos, la interface y su software controlador permitirán su adición al sistema. Se pueden conectar varios dispositivos a la vez, teniendo cada uno de ellos un



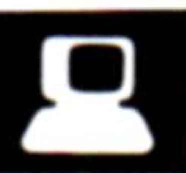
Bien avenidos
El ordenador personal MSX PX-7 y el reproductor de discos láser LD-700, de Pioneer, están diseñados para trabajar juntos. El PX-7 se puede programar utilizando ampliaciones al BASIC MSX estándar para controlar el reproductor de discos láser de modo de poder seleccionar y visualizar secuencias o fotogramas individuales. También se pueden mezclar gráficos y sonido del ordenador con la salida de video del reproductor de discos, para producir video interactivo bajo el control del ordenador

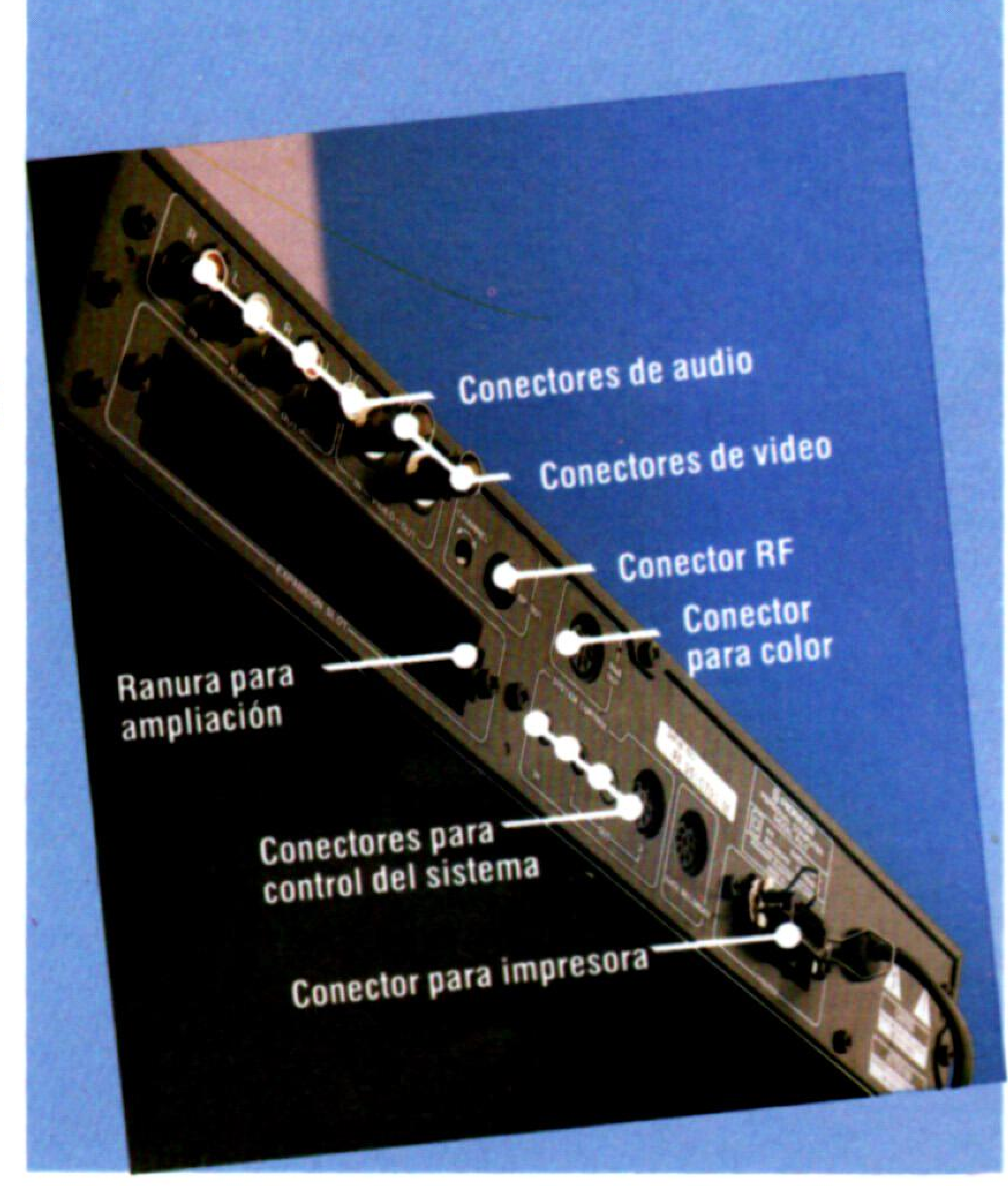


Chris Stevens



Pistas láser
Los discos láser CAV *(constant angular velocity*: velocidad angular constante), al igual que los discos magnéticos, ofrecen acceso directo a cualquier parte del disco. El mecanismo controlador puede localizar secciones individuales en la superficie del disco en relación a las pistas radiales que vemos aquí. Cada anillo del disco corresponde a un fotograma del programa almacenado, reteniendo el número de fotograma en la parte correspondiente de la pista radial





Conectores a granel
Una de las características sobresalientes del PX-7 es su amplia gama de conectores, que permiten la conexión, mediante cables estándares, a una completa gama de equipos de audio y video

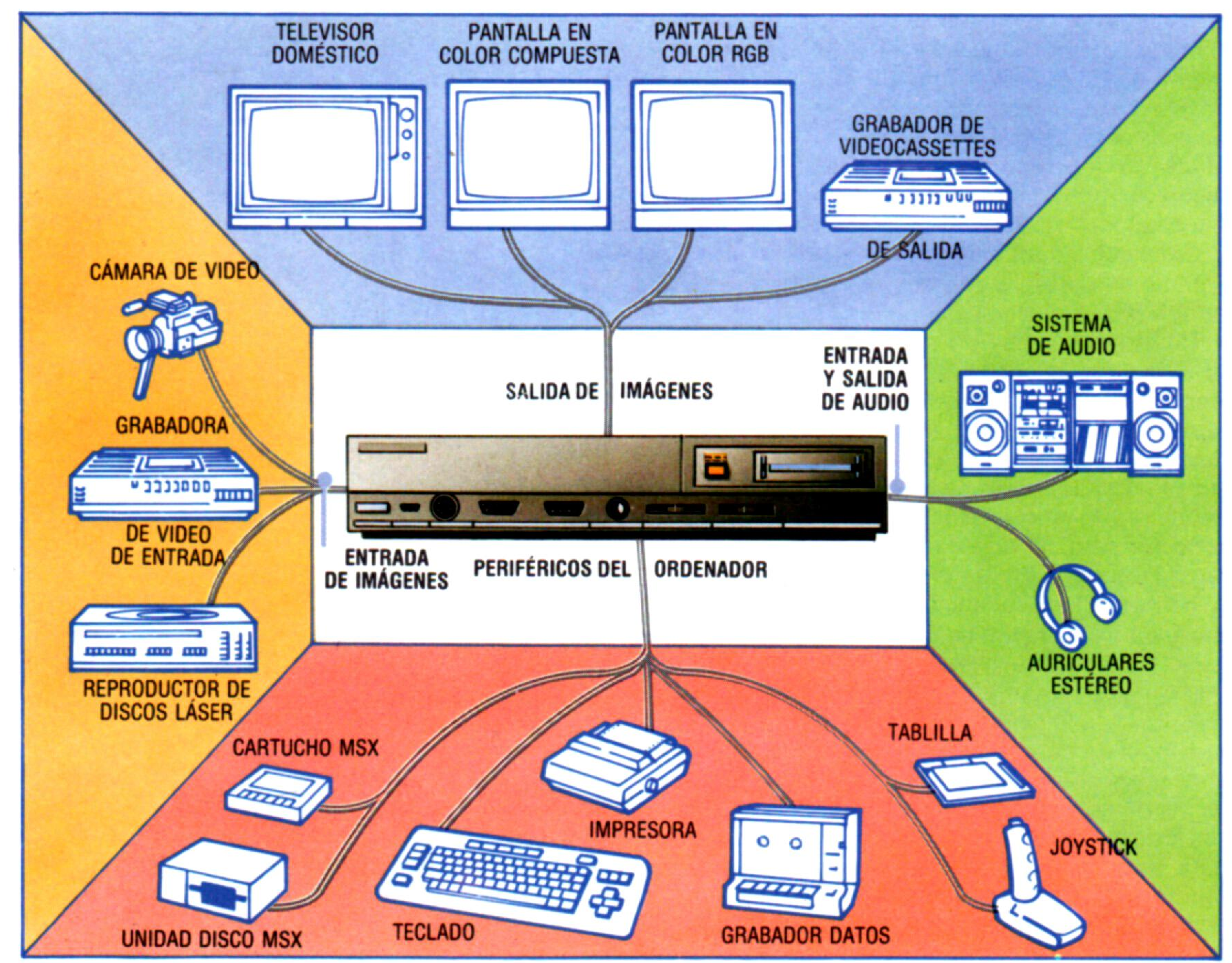
código de dispositivo para permitir que un programa especifique el dispositivo al cual está dirigida una instrucción.

El control de la interface es simple en virtud de un conjunto de ampliaciones del BASIC MSX retenidas en ROM. Cuando se pone por primera vez en marcha el ordenador, usted puede elegir ejecutar el BASIC MSX común o el P-BASIC (BASIC MSX con instrucciones para el control del sistema). Las nuevas instrucciones tienen la forma de sentencias CALL, de modo que el BASIC MSX propiamente dicho es el estándar. La mayor parte de las 16 nuevas instrucciones están relacionadas con el control de las conexiones de video y audio del ordenador. Se puede visualizar la visualización del ordenador, la visualización de video entrante (como un disco láser) o superponer ambas. Esto permite que los gráficos y el texto del ordenador aparezcan encima de las imágenes procedentes del disco láser o de la videocinta. Asimismo, puede alternar entre las modalidades ordenador, video y superposición, utilizando cuatro teclas adicionales del teclado que son las únicas adiciones hechas al MSX estándar.

Las instrucciones de sonido permiten poner sordina a uno de los canales estéreo o a ambos y regular el equilibrio entre ambos. Hay un conjunto de extras para mejorar el BASIC MSX: instrucciones que borran la pantalla de diversas maneras, guardan y cargan la visualización en una unidad de casette, etc. Sin embargo, la instrucción más importante es CALL REMOTE, que envía una instrucción a la interface de control del sistema.

Existen numerosas instrucciones específicas para el control del disco láser. Por ejemplo, CALL SEARCH (0,F,2000) pedirá al reproductor que busque en el disco el fotograma 2 000. CALL FRAME es la instrucción más sofisticada. Una vez ejecutada, la subrutina especificada se ejecutará automáticamente cuando el reproductor muestre un fotograma o capítulo determinado del disco. Esto permite vincular estrechamente un programa al sistema de disco láser. Por ejemplo, en un programa educativo, el ordenador podría preguntar ¿Quieres saber algo más acerca de esto? cada vez que un estudiante reprodujera una parte determinada del disco, pudiendo pasar a mostrar otra sección de la película.

El LD-700 es un reproductor de disco láser es-



Centro de control
Una de las intenciones originales del estándar MSX era la de permitir que los ordenadores conformaran el centro de control de un sistema de entretenimiento doméstico completo, integrado por componentes de audio y video. El Pioneer PX-7 es la primera máquina MSX que puede utilizarse de este modo. La adición del P-BASIC, una ampliación del BASIC MSX estándar, permite que el software controle verdaderas imágenes de video y sonido. Éstos se pueden mezclar con gráficos generados por ordenador, abriendo las puertas de este modo a los juegos de aventuras de video, programas de enseñanza por video interactivo y simulaciones más realistas



Ranura para cartuchos
La ranura para cartuchos permite utilizar paquetes de software basado en ROM

Altavoces
El PX-7 posee dos altavoces estéreo incorporados

ROM de P-BASIC
Esta ROM de 8 K retiene las instrucciones del P-BASIC de Pioneer que permiten controlar un disco láser desde un programa en BASIC

CPU Z80
Al igual que todos los ordenadores MSX, el sistema está construido alrededor del procesador Z80

Conector para teclado
Aquí se enchufa el teclado externo del PX-7

Puertas controladoras
A través de estos conectores D estándares de 9 patillas se puede conectar una palanca de mando *(joystick)* o una tablilla al tacto

Controles de mezcla
Estos mandos deslizables permiten establecer el volumen global y mezclar sonido externo con el generado por ordenador

Chips de sonido
Estos chips Yamaha actúan como controladores de amplificación y mezcla de sonidos

Chris Stevens

tándar y se puede adquirir y utilizar independientemente del ordenador PX-7. Cada disco de 12 pulgadas puede almacenar hasta 54 000 fotogramas, y esto significa que puede mostrar películas enteras, además de ofrecer fotogramas individuales, cámara lenta y rápida de una calidad muy superior a la que permite el video. El controlador a distancia que se suministra con la unidad permite la búsqueda de fotograma y capítulo (siendo el capítulo una división del disco). Cada disco lleva dos pistas de audio, permitiendo el almacenamiento de la banda sonora en dos idiomas, si bien algunos discos utilizan la segunda pista para almacenar un programa de ordenador ya hecho.

El hardware del Pioneer está bien construido y acabado, aunque el ordenador puede calentarse mucho al trabajar en unión con el reproductor de discos láser. Pioneer ofrece la tablilla para gráficos PXTB-7 y un paquete para gráficos basado en cartuchos denominado *Video Art*, que permite diseñar programas que utilicen gráficos de ordenador superpuestos sobre imágenes de disco láser. No obstante, el software resulta decepcionante: es lento, difícil de emplear y de capacidades limitadas.

El PX-7 hace que la creación y ejecución de programas de video interactivos para el hogar, el despacho o la clase resulten muy sencillas. Escribir software adecuado utilizando las ampliaciones incorporadas del BASIC es extraordinariamente fácil, y el propio BASIC MSX es sofisticado y moderadamente veloz. La velocidad no constituye mayor problema, puesto que el reproductor tarda alrededor de dos segundos en buscar un fotograma determinado.

En vez de quedar limitados a un sistema económico, el PX-7 y el LD-700 proporcionan el potencial para programas de video interactivos completos. Usted podría producir juegos de disco láser, como los que han obtenido tanto éxito en las salas recreativas, crear una gran base de datos de imágenes, etc. El sistema también acepta discos CPE *(computer program encoded*: con programa para ordenador codificado), es decir, aquellos que en una de las dos pistas de audio tienen almacenado un programa para ordenador ya listo.

Sin embargo, es probable que el costo de los discos matriz impida que muchas personas diseñen sus propios discos, y los usuarios habrán de confiar en productos comerciales de empresas cinematográficas o de software. Como es habitual con la nueva tecnología, la aparición del software que explote cabalmente el nuevo hardware llevará su tiempo. Mientras tanto, el Pioneer PX-7 es uno de los más interesantes ordenadores MSX y, en unión de un disco láser, representa cabalmente el futuro de la informática del ocio doméstico y de la educación.

PIONEER PX-7

DIMENSIONES

420 × 323 × 70 mm

MEMORIA

32 K de RAM para el usuario, ampliables a 64 K; 16 K de RAM de video, 40 K de ROM

CPU

Procesador Z80 a 4 MHz

PANTALLA

40 columnas × 24 líneas, 16 colores, gráficos en alta resolución de 256 × 192, con hasta 32 sprites utilizando el chip de visualización 9929

SONIDO

Salidas estéreo y sonido a tres voces utilizando el chip de sonido 8910

INTERFACES

Video compuesto, TV, RGB, impresora Centronics, 2 puertas para palanca de mando *joystick* y 2 puertas para cartuchos, cassette, entrada audio, salida audio, auriculares, control del sistema

LENGUAJES DISPONIBLES

BASIC MSX 32 K y P-BASIC 8 K

DOCUMENTACION

El PX-7 utiliza pequeños folletos, como los que cabría esperar en un sistema *hi-fi*. Algunos están muy pobremente traducidos del japonés y toda la información se presenta en un anodino estilo de informe. Los manuales, sin embargo, son completos e incluyen gran cantidad de información técnica

VENTAJAS

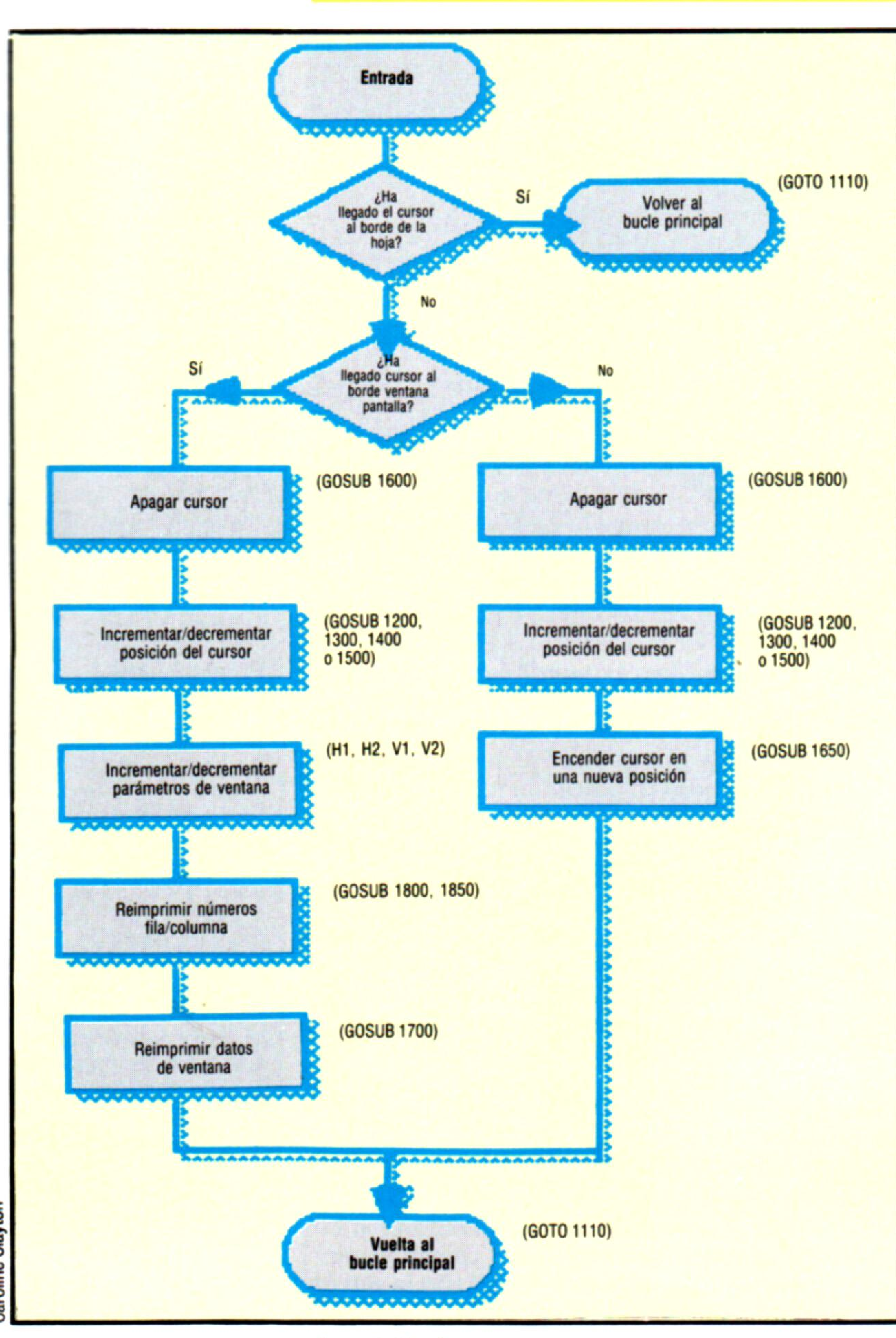
El PX-7 es el más desarrollado de todos los sistemas MSX disponibles. En unión del disco láser, representa un auténtico avance en la tecnología del entretenimiento doméstico. Las unidades están bien construidas, su diseño es agradable y su precio, moderado

DESVENTAJAS

MSX ha recibido un mínimo apoyo por parte de empresas independientes. Asimismo, el PX-7 sufre las consecuencias de la escasez de software y discos láser para sacar partido de sus facilidades

Cuadrícula inicial

Esta vez nos corresponde programar las visualizaciones en pantalla para el Amstrad CPC 464/664, el BBC Micro y el Sinclair Spectrum



Cerca del borde
El diagrama de flujo muestra la manera en que nuestro programa controla al cursor sobre la «hoja». Dado que la pantalla sólo es una ventana de la hoja electrónica, cuando el cursor llega a los bordes superior, inferior, izquierdo o derecho de la zona de pantalla visible, se emprenderá una acción para pasar a la siguiente porción de la hoja

Debido a que los micros para los que se ha diseñado la hoja electrónica poseen métodos diferentes para producir visualizaciones en pantalla, ofrecemos las rutinas gráficas para cada uno de forma independiente. (Anteriormente se ha ilustrado la versión para el Commodore 64.) Las principales funciones de esta parte del programa son imprimir en la pantalla la cuadrícula de la hoja electrónica, junto con los números de filas y columnas, y controlar el movimiento del cursor de la hoja electrónica a través de la cuadrícula.

En las rutinas de manipulación del cursor se incluyen secciones que desplazan el cursor a izquierda, derecha, arriba y abajo. La pantalla sólo puede visualizar en cada momento una porción de la hoja electrónica (le será útil pensar en la pantalla como una ventana a través de la cual sólo se ve una parte de la hoja) y, por tanto, las rutinas del cursor también manipularán el movimiento de la ventana a través de la hoja electrónica.

Las líneas 1000-1080 conforman una subrutina que imprime la cuadrícula de la hoja electrónica. La siguiente sección, que empieza en la línea 1100, es el bucle de control principal del programa, que esencialmente explora el teclado en busca de una pulsación de tecla. Se pueden utilizar teclas para trasladar el cursor por la hoja o para seleccionar una función, como entrar una fórmula en una celda determinada. Desde esta sección se llama la subrutina adecuada.

Movimiento del cursor

La mayoría de las funciones de la hoja electrónica serán tema de futuros capítulos y, por lo tanto, intentar seleccionarlas en esta etapa sólo hará que el programa quede colgado. No obstante, las subrutinas incluidas en este capítulo (en las líneas 1200, 1300, 1400 y 1500) le permitirán desplazar el cursor por la pantalla, puesto que son las rutinas de movimiento del cursor para DERECHA, IZQUIERDA, ABAJO y ARRIBA, respectivamente. Estas rutinas son similares, de modo que sólo veremos una de ellas para hacernos una idea del modo en que trabajan las cuatro.

Al pulsar la tecla para mover el cursor hacia la derecha, el programa salta a la subrutina de la línea 1200, y la línea 1210 comprueba si el cursor ha llegado al borde de la hoja. Esto lo realiza comprobando la coordenada X, para determinar si ha alcanzado su valor máximo, 15. De ser así, se devuelve el control al bucle principal del programa; de lo contrario, la rutina sigue adelante. El siguiente paso consiste en ver si el cursor se halla en el extremo derecho de la ventana de pantalla actual. Se utilizan las variables H1 y H2 para los límites horizontales inferior y superior de la ventana; por ejemplo, si H1=2 y H2=6, en la pantalla están visibles las columnas 2 a 6 de la hoja. Si X ha alcanzado el valor retenido en H2, se pone en movimiento la



siguiente cadena de acontecimientos: se apaga el cursor, se incrementa el valor de X, se incrementan los valores de H1 y H2, la subrutina de la línea 1800 imprime nuevos números de fila y columna, y se imprimen en la hoja los datos de la nueva celda. Por último, se vuelve a encender el cursor.

Las subrutinas de las líneas 1600 y 1650 son necesariamente diferentes para cada uno de los cuatro ordenadores, y controlan el encendido y el apagado del cursor.

En todas las versiones, éste se muestra invirtiendo los colores del primer plano y del fondo de la celda adecuada, pero la forma en que se consigue este efecto es peculiar del hardware y el firmware de visualización de cada máquina.

En el Spectrum, los colores del primer plano y del fondo de cada celda de caracteres son controlados por un byte de la memoria llamado *mapa de atributos*. Los tres bits inferiores del byte de atributos controlan el color INK y los bits tres a cinco controlan el color PAPER. En consecuencia, para invertir los colores sólo es necesario localizar el grupo de bytes de atributos que corresponde a la celda actual de la hoja electrónica y colocar (POKE) los valores apropiados.

En las versiones para el Amstrad y el BBC Micro, el proceso es ligeramente más complicado, porque el valor de la celda se debe reimprimir en la celda después de encender o apagar el cursor. Los valores de la celda están retenidos en la matriz M(,); se debe hallar el valor correcto para la celda actual y convertirlo en una serie lista para impresión. En el BBC Micro, los colores se pueden intercambiar simultáneamente utilizando la instrucción COLOUR. En el Amstrad hay disponible un carácter de control, CHR$(24), que se puede incorporar a una sentencia PRINT para intercambiar los colores PEN y PAPER actuales.

Sinclair Spectrum:

```
100 GO SUB 3000
110 GO SUB 1000
120 GO SUB 1700
130 GO SUB 1100
999 STOP
1000 BORDER 1: PAPER 1: CLS : INK 7
1005 PRINT "        C O L U M N A S"
1007 PRINT "FILA     1.    2.    3.    4."
1010 PRINT "-------+-----+-----+-----+-----+"
1020 FOR C=1 TO 6
1030 PRINT CHR$ (C+64);".      |     |
     |     |     |"
1040 PRINT "-------+-----+-----+-----+-----+"
1050 NEXT C
1060 PRINT CHR$ (C+64);".      |     |
     |     |     |"
1070 PRINT "-------+-----+-----+-----+-----+"
1080 RETURN
1100 PRINT AT 0,0;"CELDA:";CHR$(Y+64);STR$
     (X)
1110 LET A$=INKEY$: IF A$="" THEN GO TO
     1110
1120 IF A$="8" THEN GO TO 1200: REM MUEVE
     DERECHA
1130 IF A$="5" THEN GO TO 1300: REM MUEVE
     IZQUIERDA
1140 IF A$="6" THEN GO TO 1400: REM MUEVE
     ABAJO
1150 IF A$="7" THEN GO TO 1500: REM MUEVE
     ARRIBA
1155 IF A$="C" THEN GO SUB 2300: REM
     CALCULA HOJA
1160 IF A$="F" THEN GO SUB 2000: REM ENTRA
     FORMULA
1165 IF A$="E" THEN GO SUB 2100: REM ENTRA
     DATOS NUMERICOS EN CELDA
1168 IF A$="H" THEN GO TO 6000: REM
     IMPRIME PANTALLA AYUDA
1170 IF A$="S" THEN GO SUB 5150: REM
     RECUPERA HOJA ACTUAL EN MEMORIA
1172 IF A$="G" THEN GO SUB 5100: REM TOMA
     HOJA ANTERIOR
1174 IF A$ "Z" THEN GO SUB 5000: REM BORRA
     HOJA ACTUAL
1176 IF A$="R" THEN GO SUB 5700: REM
     REPRODUCE FORMULA
1178 IF A$="T" THEN GO SUB 5200: REM TAB A
     NUEVA CELDA
1180 IF A$="D" THEN GO SUB 7000: REM
     CARGA/GUARDA DATOS/FORMULAS
1190 GO TO 1110
1200 REM ***** MUEVE DERECHA *****
1210 IF X=15 THEN GO TO 1100
1220 IF X=H2 THEN GO SUB 1600: LET X=X+1:
     LET H1=H1 + 1: LET H2=H2 + 1: GO TO
     1270
1230 GO SUB 1600: LET X=X+1: GO SUB 1650:
     GO TO 1100
1270 GO SUB 1800: GO SUB 1700: GO TO 1100
1300 REM ***** MUEVE IZQUIERDA *****
1310 IF X=1 THEN GO TO 1100
1320 IF X=H1 THEN GO SUB 1600: LET X=X–1:
     LET H1=H1–1: LET H2=H2–1: GO TO 1370
1330 GO SUB 1600: LET X=X–1: GO SUB 1650:
     GO TO 1100
1370 GO SUB 1800: GO SUB 1700: GO TO 1100
1400 REM ***** MUEVE ABAJO *****
1410 IF Y=15 THEN GO TO 1100
1420 IF Y=V2 THEN GO SUB 1600: LET Y=Y+1:
     LET V1=V1+1: LET V2=V2+1: GO TO 1470
1430 GO SUB 1600: LET Y=Y+1: GO SUB 1650:
     GO TO 1100
1470 GO SUB 1850: GO SUB 1700: GO TO 1100
1500 REM ***** MUEVE ARRIBA *****
1510 IF Y=1 THEN GO TO 1100
1520 IF Y=V1 THEN GO SUB 1600: LET Y=Y1–1:
     LET V1=V1–1:LET V2=V2–1: GO TO 1570
1530 GO SUB 1600: LET Y=Y–1: GO SUB 1650:
     GO TO 1100
1570 GO SUB 1850: GO SUB 1700: GO TO 1100
1600 REM ***** APAGAR CURSOR *****
1610 LET CU=22528+32*(V(Y+1–V1))
     +H(X+1–H1)
1615 POKE CU,15: POKE CU+1,15: POKE
     CU+2,15
1620 POKE CU+3,15: POKE CU+4,15: RETURN
1650 REM ***** ENCIENDE CURSOR *****
1660 LET CU=22528+32*(V(Y+1–V1))
     +H(X+1–H1)
1665 POKE CU,56: POKE CU+1,56: POKE
     CU+2,56
1670 POKE CU+3,56: POKE CU+4,56
1690 GO SUB 1900: RETURN
1700 REM ***** IMPRIME DATOS EN HOJA
     *****
1710 FOR I=0 TO 6
1720 FOR J=0 TO 3
1730 LET P$=STR$ (M(I+V1,J+H1))
1740 PRINT AT V(I+1),H(J+1);"     "
1745 PRINT AT V(I+1),(H(J+1)+5-LEN (P$));P$
1750 NEXT J: NEXT I
1760 GO SUB 1650: RETURN
1800 REM ***** IMPRIME NUMS
     COLUMNAS*****
1810 FOR I=H1 TO H2
1820 PRINT AT 1,7+6*(I–H1);I;"."
1830 NEXT I
1840 RETURN
1850 REM *** IMPRIME NUMS FILAS ***
1870 FOR C=V1 TO V2
1880 PRINT AT 2*(C–V1)+3,0;
     CHR$(C+4);"."
1890 NEXT C
1895 RETURN
1900 REM * FORMULA CELDA ACTUAL *
1920 LET D$=F$((Y–1)*15+X,1 TO )
1930 PRINT AT 18,0;"FORMULA: "
1940 PRINT AT 18,0;"FORMULA: ";D$
1945 RETURN
```

Rutinas preparación matrices:

```
3000 REM ******************************
3001 REM *       PREPARA MATRICES     *
3002 REM ******************************
3010 DIM H(4): DIM V(7): DIM S(20): DIM S$(20):
     DIM G$ (20): DIM C(20)
3020 FOR C=0 TO 3
3030 LET H(C+1)=6*C+8: REM CALC POS–X
3040 NEXT C
3050 FOR C=1 TO 7
3060 LET V(C)=2*C+1: REM CALC POS–Y
3070 NEXT C
3075 LET X=1: LET Y=1
3080 LET H1=X: LET H2=X+3: LET V1=1: LET
     V2=V1+6
3090 REM ******************************
3091 REM *       MATRIZ VALORES       *
3092 REM ******************************
3100 DIM M(15,15): DIM N(15,15)
3110 FOR I=1 TO 15
3120 FOR J=1 TO 15
3130 LET M(I,J)=I*J
3140 NEXT J: NEXT I
3150 DIM F$(255,20): DIM G$(20): RETURN
```

BBC Micro:



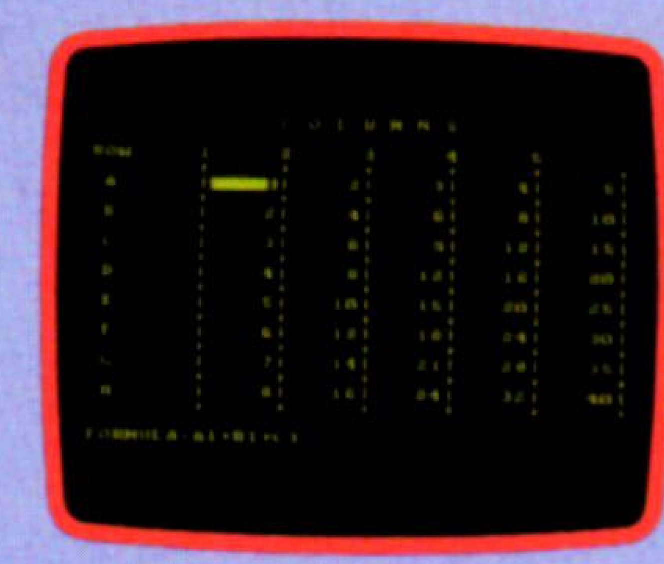
```
10 REM **** HOJA ELECTRONICA BBC ****
40 MODE 4
50 *FX4,1
60 LET CO$=CHR$(30):CL$=CHR$(8):CR$=
   CHR$(9):CU$=CHR$(11): CD$=CHR$(10)
70 REM VDU 23,1,0;0;0;0;
100 GOSUB 3000:REM PREPARA MATRICES Y VARIABLES
    PANTALLA
110 GOSUB 1000:REM IMPRIME PANTALLA
120 GOSUB 1700:REM IMPRIME VENTANA DATOS EN
    PANTALLA
130 GOTO 1100:REM RUTINA PRINCIPAL TECLADO
999 STOP
1000 PRINT CHR$(12)
1005 PRINT "                    C O L U M N A S"
1006 PRINT
1007 PRINT "FILA          1.    2.    3.    4.    5."
1010 PRINT "           +-----+-----+-----+-----+-----+"
1020 FOR C=1 TO 7
1030 PRINT " "CHR$(C+64);".  |     |     |     |     |     |"
1040 PRINT "-------+-----+-----+-----+-----+-----+"
1050 NEXT G
1060 PRINT " "CHR$(C+64);".  |     |     |     |     |     |"
1070 PRINT "-------+-----+-----+-----+-----+-----+"
1080 RETURN
1100 P$=CHR$(Y+64)+STR$(X):PRINT
     CO$;CD$"CELDA:";P$;" "
1110 LET A$=GET$
1120 LET A$=CHR$(137) THEN 1200:REM MUEVE CURSOR
     DERECHA
1130 IF A$=CHR$(136) THEN 1300:REM MUEVE CURSOR
     IZQUIERDA
1140 IF A$=CHR$(138) THEN 1400:REM MUEVE CURSOR ABAJO
1150 IF A$=CHR$(139) THEN 1500:REM MUEVE CURSOR
     ARRIBA
1152 IF A$="H" THEN GOSUB 6000:REM IMPRIME PANTALLA
     AYUDA
1154 IF A$="F" THEN GOSUB 2000:REM ENTRA FORMULA
1156 IF A$="S" THEN GOSUB 5150:REM ALMACENA HOJA
     ACTUAL
1158 IF A$="G" THEN GOSUB 5100:REM TOMA HOJA
     ANTERIOR
1160 IF A$="C" THEN GOSUB 2300:REM CALCULA HOJA
1165 IF A$=CHR$(13) THEN RETURN
1170 IF A$>="0" AND A$<="9" THEN GOSUB 2100:REM
     RUTINA ENTRADA DATOS
1180 IF A$="Z" THEN GOSUB 5000:REM BORRAR HOJA
1185 IF A$="R" THEN GOSUB 5700:REM DUPLICA HOJA
1187 IF A$="T" THEN GOSUB 5200: REM TAB A NUEVA CELDA
1189 IF (INKEY(-119)) THEN GOSUB 7000:REM RUTINAS
     CARGAR/GUARDAR
1190 GOTO 1100:REM VUELVE AL COMIENZO
1200 REM ***** MUEVE DERECHA *****
1210 IF X=15 THEN 1100
1220 IF X=H2 THEN GOSUB 1600:X=X+1:H1=
     H1+1:H2=H2+1:GOTO 1270
1230 GOSUB 1600:LET X=X+1:GOSUB 1650:GO TO 1100
1270 GOSUB 1800:GOSUB 1700:GOTO 1100
1300 REM ****** MUEVE IZQUIERDA ******
1310 IF X=1 THEN 1100
1320 IF X=H1 THEN GOSUB 1600:X=X-1:H1=
     H1-1:H2=H2-1:GOTO 1370
1330 GOSUB 1600:LET X=X-1:GOSUB 1650:GOTO 1100
1370 GOSUB 1800:GOSUB 1700:GOTO 1100
1400 REM ***** MOVER ABAJO *****
1410 IF Y=15 THEN 1100
1420 IF Y=V2 THEN GOSUB 1600:LET Y=Y+1:V1=
     V1+1:V2=V2+1:GOTO 1470
1430 GOSUB 1600:LET Y=Y+1:GOSUB 1650:GOTO 1100
1470 GOSUB 1850:GOSUB 1700:GOTO 1100
1500 REM ***** MUEVE ARRIBA *****
1510 IF Y=1 THEN 1100
1520 IF Y=V1 THEN GOSUB 1600:LET Y=Y-1:V1=
     V1-1:V2=V2-1:GOTO 1570
1530 GOSUB 1600:LET Y=Y-1:GOSUB 1650:GOTO 1100
1570 GOSUB 1850:GOSUB 1700:GOTO 1100
1600 REM ***** APAGA CURSOR *****
1610 P$=STR$(M(Y-V1+1,X-H1+1))
1615 IF LEN(P$)< 5 THEN P$=" "+P$:GOTO 1615
1620 COLOUR 1:COLOUR 128
1625 PRINT TAB(H(X-H1+1)-1,V(Y-V1+1)-1);P$
1630 COLOUR 1: COLOUR 128
1640 RETURN
1650 REM ***** ENCIENDE CURSOR *****
1660 P$=STR$(M(Y-V1+1,X-H1+1))
1665 IF LEN(P$)< 5 THEN P$=" "+P$:GOTO 1665
1670 COLOUR 2:COLOUR 129
1675 PRINT TAB(H(X-H1+1)-1,V(Y-V1+1)-1;P$
1680 COLOUR 1: COLOUR 128
1690 GOSUB 1900:RETURN
1700 REM ***** IMPRIME VENTANA DATOS DESDE HOJA EN
     PANTALLA *****
1710 FOR I=0 TO 7
1720 FOR J=0 TO 4
1730 P$=STR$(M(I+V1,J+H1))
1735 IF LEN(P$)<5 THEN P$=" "+P$:GOTO 1735
1740 PRINT TAB(H(J+1)-1,V(I+1)-1);"      "
1745 PRINT TAB(H(J+1)-1,V(I+1)-1);P$
1750 NEXT J, I
1760 GOSUB 1650:RETURN
1800 REM ***** IMPRIME NUM COLUMNA *****
1810 FOR I=H1 TO H2:PRINT TAB(8+6*(I-H1),3);I;"."
1820 NEXT I:RETURN
1850 REM ***** IMPRIME ETIQUETAS FILAS *****
1860 FOR C=V1 TO V2
1870 PRINT TAB(1,V(C-V1+1)-1);CHR$(C+64);"."
1880 PRINT: NEXT C: RETURN
1900 REM ***** FORMULA DE CELDA ACTUAL *****
1910 LET D$=F$((Y-1)*15+X)
1920 PRINT TAB(0,22);"    "
1930 PRINT TAB(0,22);"FORMULA: ";D$
1940 RETURN
```

Rutinas preparación matrices:

```
3000 REM ***** PREPARAR MATRICES *****
3010 DIM H(5),V(8),ST(20),ST$(20),E$(20),G$(20),C(20)
3020 FOR C=0 TO 4
3030 LET H(C+1)=6*C+10
3040 NEXT C
3050 FOR C=1 TO 8
3060 LET V(C)=2*C+4:REM CALC POS Y
3070 NEXT C
3075 X=1:Y=1:REM POSICION INICIAL CURSOR
3080 H1=X:H2=X+4:V1=Y:V2=Y+7
3090 REM ***** DIM MATRICES HOJA *****
3100 DIM M(15,15):DIM N(15,15)
3110 FOR I=1 TO 15:FOR J=1 TO 15
3120 LET M(I,J)=I*J+I
3130 NEXT J,I
3140 DIM F$(225)
3150 RETURN
```



Amstrad CPC 464/664:



```
100 GOSUB 3000:REM PREPARA MATRICES Y VARIABLES
110 GOSUB 1000:REM IMPRIME PANTALLA
120 GOSUB 1700:REM IMPRIME DATOS EN PANTALLA
130 GOTO 1100
999 STOP
1000 REM IMPRIME VISUALIZACION PANTALLA
1005 CLS:PRINT "                    C O L U M N A S"
1006 PRINT
1007 PRINT "FILA          1.    2.    3.    4.    5."
1010 PRINT "          +-----+-----+-----+-----+-----+"
1020 FOR C=1 TO 7
1030 PRINT " "CHR$(C+64);".  |     |     |     |     |     |"
1040 PRINT " -------+-----+-----+-----+-----+-----+"
1050 NEXT C
1060 PRINT " "CHR$(C+64);".  |     |     |     |     |     |"
1070 PRINT " ------+-----+-----+-----+-----+-----+"
1080 RETURN
1100 LOCATE 1,1:P$=CHR$(Y+64)+MID$(STR$(X),2,2):PRINT
     "CELDA "; P$;" "
1110 A$=INKEY$:IF A$="" THEN 1110
1120 IF A$=CHR$(243) THEN 1200:REM MUEVE DERECHA
1130 IF A$=CHR$(242) THEN 1300:REM MUEVE IZQUIERDA
1140 IF A$=CHR$(241) THEN 1400:REM MUEVE ABAJO
1150 IF A$=CHR$(240) THEN 1500:REM MUEVE ARRIBA
1155 IF A$="F" THEN GOSUB 2000:REM ENTRA FORMULA
1160 IF A$="C" THEN GOSUB 2300:REM CALCULA HOJA
1165 IF A$=CHR$(13) THEN RETURN
1170 IF A$>="0" AND A$<"9" THEN GOSUB 2100:REM ENTRA
     DATOS NUMERICOS
1180 IF A$="B" THEN GOSUB 5000:REM BORRA HOJA
1185 IF A$="V" THEN GOSUB 5100:REM TOMA HOJA ANTERIOR
1187 IF A$="G" THEN 5200:REM MUEVE CURSOR A NUEVA
     CELDA
1188 IF A$="R" THEN GOSUB 5700
1190 GOTO 1100
1200 REM ***** MUEVE DERECHA *****
1210 IF X=15 THEN 1100
1220 IF X=H2 THEN GOSUB 1600:X=X+1:H1=
     H1+1:H2=H2+1:GOTO 1270
1230 GOSUB 1600:LET X=X+1:GOSUB 1650:GOTO 1100
1270 GOSUB 1800:GOSUB 1700:GOTO 1100
1300 REM ***** MUEVE IZQUIERDA *****
1310 IF X=1 THEN 1100
1320 IF X=H1 THEN GOSUB 1600:X=X-1:H1=
     H1-1:H2=H2-1:GOTO 1370
1330 GOSUB 1600:LET X=X-1:GOSUB 1650:GOTO 1100
1370 GOSUB 1800:GOSUB 1700:GOTO 1100
1400 REM ***** MUEVE ABAJO *****
1410 IF Y=15 THEN 1100
1420 IF Y=V2 THEN GOSUB 1600:Y=Y+1:V1=
     V1+1:V2=V2+1:GOTO 1470
1430 GOSUB 1600:LET Y=Y+1:GOSUB 1650:GOTO 1100
1470 GOSUB 1850:GOSUB 1700:GOTO 1100
1500 REM ***** MUEVE ABAJO *****
1510 IF Y=1 THEN 1100
1520 IF Y=V1 THEN GOSUB 1600:Y=Y-1:V1=
     V1-1:V2=V2-1:GOTO 1570
1530 GOSUB 1600:LET Y=Y-1:GOSUB 1650:GOTO 1100
1570 GOSUB 1850:GOSUB 1700:GOTO 1100
1600 REM **** APAGA CURSOR ****
1610 LET P$=MID$(STR$(M(Y,X)),2)
1615 IF LEN(P$)<5 THEN P$=" "+P$:GOTO 1615
1620 LOCATE H(X-H1+1)-1,V(Y-V1+1):PRINT P$
1630 RETURN
1650 REM *** ENCIENDE CURSOR ***
1660 LET P$=MID$(STR$(M(Y,X)),2)
1665 IF LEN(P$)<5 THEN P$=" "+P$:GOTO 1665
1670 LOCATE H(X-H1+1)-1,V(Y-V1+1):PRINT
     CHR$(24);P$;CHR$(24);
1680 GOSUB 1900:RETURN
1700 REM **** IMPRIME DATOS EN HOJA ****
1710 FOR I=0 TO 7
1720 LOCATE 10,V(I+1)
1730 FOR J=0 TO 4
1735 LET P$=MID$(STR$(M(I+V1,J+H1)),2)
1740 IF LEN(P$)<5 THEN P$=" "+P$:GOTO 1740
1745 LOCATE H(J+1)-1,V(I+1):PRINT P$;
1750 NEXT J,I
1760 GOSUB 1650:REM ENCIENDE CURSOR
1770 RETURN
1800 REM ***** IMPRIME NUMS COLUMNAS *****
1810 LOCATE 1,3:PRINT "FILA    ";;
1820 LOCATE 7,3:FOR I=H1 TO H2:PRINT TAB(H(I-H1+1)
     -3)I;CHR$(8);".";
1830 NEXT I
1840 RETURN
1850 REM ***** IMPRIME LETRAS FILAS *****
1860 LOCATE 1,4
1870 FOR I=V1 TO V2
1875 PRINT
1880 PRINT " ";CHR$(I+64);"."
1890 NEXT I
1895 RETURN
1900 REM ***** FORMULA DE CELDA ACTUAL *****
1920 LET D$=F$((Y-1)*15+X)
1930 LOCATE 1,22
1940 PRINT "FORMULA:
                "

1950 PRINT '


FORMULA: ";D$
1960 LOCATE 1,1
1970 RETURN
```

Rutinas preparación matrices:

```
3000 REM ******************************************
3001 REM *      PREPARA MATRICES Y VARIABLES      *
3002 REM ******************************************
3010 DIM H(5),V(8),ST(20),ST$(20),E$(20),G$(20),C(20)
3020 FOR C=0 TO 4
3030 H(C+1)=6*C+11:REM CALC POS X
3040 NEXT C
3050 FOR C=1 TO 8
3060 LET V(C)=2*C+3:REM CALC POS Y
3070 NEXT C
3075 LET X=1:Y=1
3080 LET H1=X:H2=X+4:V1=Y:V2=Y+7
3090 REM ******************************************
3091 REM *              MATRIZ VALOR              *
3092 REM ******************************************
3100 DIM M(15,15):DIM N(15,15)
3110 FOR I=1 TO 15
3120 FOR J=1 TO 15
3130 LET M(I,J)=I*J
3140 NEXT J,I
3150 REM *             MATRIZ FORMULA             *
3152 REM ******************************************
3160 DIM F$(255)
3170 LET F$(1)="A1+B1+C1"
3180 LET F$(31)="C1+C2+C3"
3190 LET F$(16)="B1+B2+B3"
3200 RETURN
```

Programa objeto

Proporcionamos las rutinas para manipular los objetos en nuestro juego «Dog and Bucket»

Nuestro árbol de manipulación de objetos se puede programar como se indica, añadiendo las correspondientes líneas a los listados centrales ya proporcionados. Aquí las líneas clave son las 210 y 220, 2430 y 2440, y 5030 a 5090. Examinemos cada uno de estos trozos clave de código de uno en uno.

Primero, las líneas 210 y 220 preparan las matrices necesarias para almacenar los datos del árbol. Recuerde que, a diferencia de algunos de los árboles anteriores que examinamos en esta serie, éste comprueba muchas condiciones diferentes, con independencia del nivel de descenso o del número de nudo. En consecuencia, necesitamos almacenar para cada nudo un registro del valor condicional que está probando y los nudos a los que conducirá, según que las condiciones resulten verdaderas o falsas. La matriz c retiene los distintos valores condicionales, y cada nudo comprueba un elemento de esa matriz. El número del elemento a comprobar se lee en la matriz k(número de árboles, número máx. de nudos de elección).

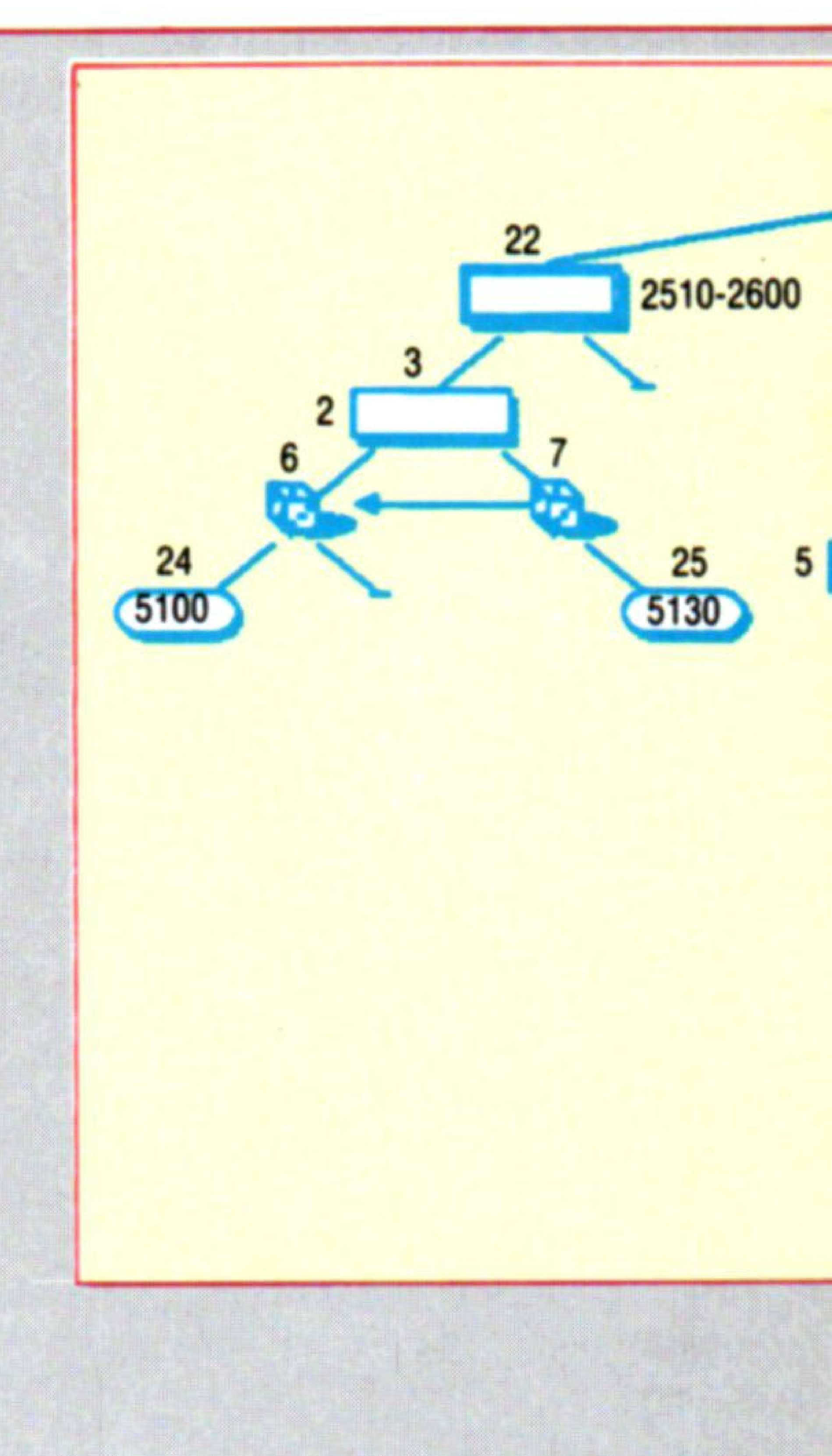
Las líneas 2430 y 2440 inicializan la matriz c. Estas líneas constituyen una subrutina que se debe llamar para cada personaje, dado que obviamente el valor de las condiciones variará para cada caso.

Entre las líneas 5030 y 5060 recorremos el árbol. La línea 5040 comprueba el número de nudo actual, y si se trata de un nudo terminal (es decir, si su numeración es mayor que 21) entonces salta fuera del árbol para seleccionar la rutina de las líneas 5070 a 5090. La línea 5050 comprueba si el nudo está probando la condición 12, que indica un nudo aleatorio, y si es así, llama la rutina de número aleatorio para asignarle un valor (uno o dos) a la condición. Luego la línea 5060 lleva a cabo la parte más importante de la operación, seleccionando el nuevo número de nudo de la matriz t y volviendo a saltar luego a la línea 5040.

La ejecución del programa completo le mostrará al manipulador de personajes en plena acción. Entre S en respuesta a la indicación ¿Valores por defecto? y vea qué sucede. El editor de personajes de la línea 2350 no es totalmente compatible con el manipulador de personajes tal como está. Esta aplicación la analizaremos con mayor detalle en el próximo capítulo. Usted puede cambiar de escenario pulsado 1, 2 o 3.

En esta etapa encontrará que la acción es más bien repetitiva, pero esta situación cambiará en seguida cuando agreguemos las dos rutinas finales (las rutinas de *interacción* y de *trazado*) en los capítulos venideros.

Al módulo de inicialización se le deben añadir las siguientes líneas. Las funciones son útiles para manipular los datos retenidos en las dos matrices de serie principales. La línea 190 prepara una matriz, *t*, que se utilizará para almacenar los datos de nuestras tres principales estructuras arborescentes: el árbol de «objetos» (en este capítulo), el árbol de *interacción* y el árbol de *trazado* (de los que hablaremos en el próximo capítulo). La matriz *k* retiene las distintas condiciones que se deben comprobar en los diversos nudos, y la matriz *c* retiene los valores condicionales (que se inicializan en las líneas 2430 y 2440)



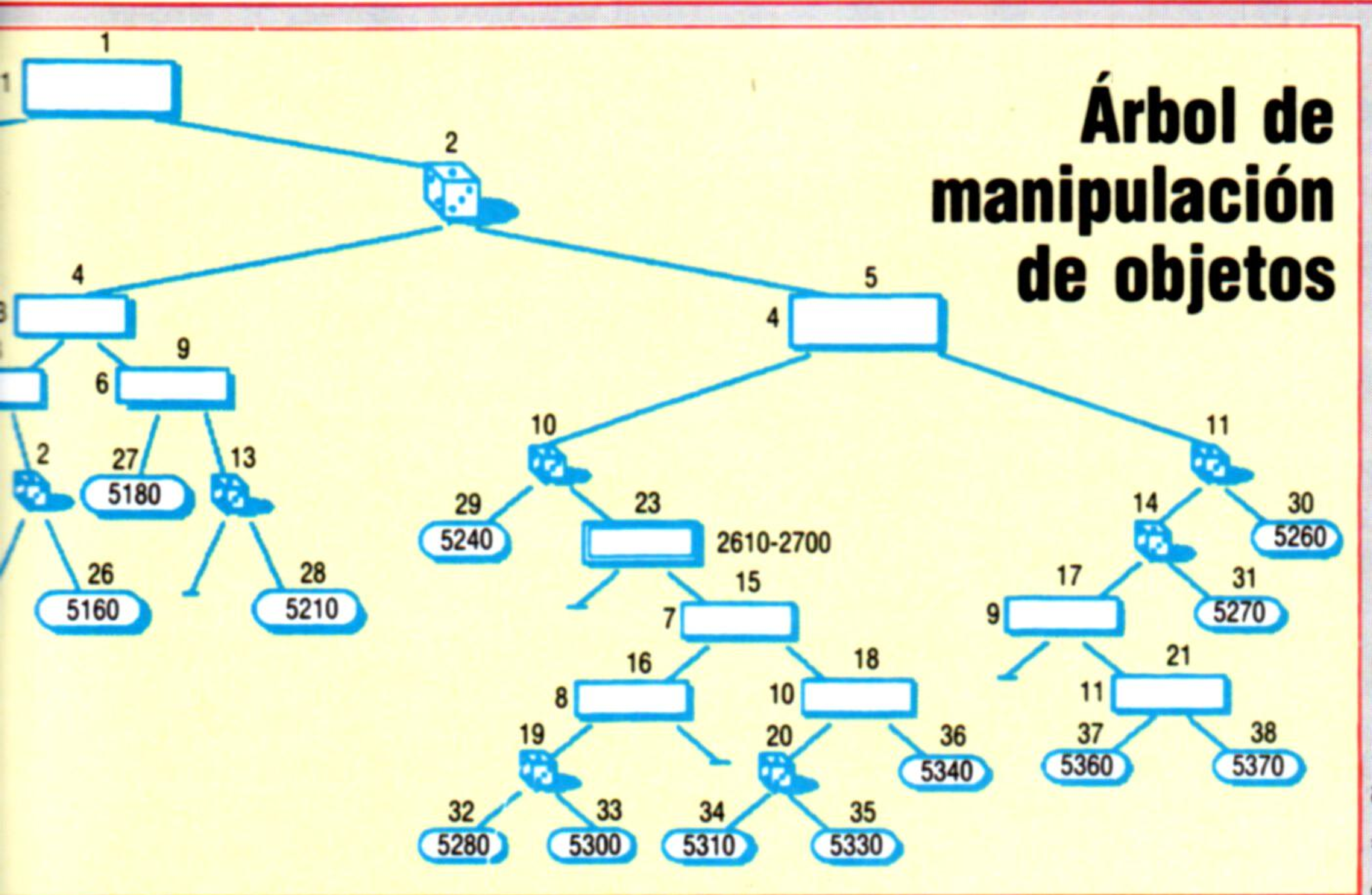

```
130 DEF FNb(y,z)=VAL(b$(y,z))
140 DEF FNc(y,z)=VAL(c$(y,z))
150 DEF FNm$(c$,d)=STR$((VAL(c$))–d)
160 DEF FNi$=b$(VAL(c$(c,3)),1)
180 REM prepara árboles
190 DIM t(3,25,2),k(3,30),c(25)
200 REM árbol objeto
210 FOR n=1 TO 21:REM 21 nudos de elección
220 READ k(1,n),t(1,n,2),t(1,n,1):NEXT n
```

Necesitamos ajustar nuestro bucle principal del programa para tener en cuenta la primera parte de nuestro manipulador de personajes. Las líneas 550 a 800 toman cada personaje de a uno, comprueban si la bandera *manipular* es mayor que cero (línea 560) y, si lo es, la reducen en uno y prosiguen al siguiente personaje. Si el valor es cero, entonces se restablece el factor de manipulación de personaje (c$(n,10)) mediante la lectura de los datos en las líneas 6030 y 6040. La línea 580 comprueba si el valor de manipulación por defecto es cero, en cuyo caso el manipulador no procesará en absoluto al personaje. La línea 590 inicializa las condiciones para el árbol llamando la subrutina de 2430, y después llama el manipulador de la línea 5000.

```
500 REM
510 REM prueba bucle programa
520 REM
530 GOSUB 2100: GOSUB 2150: GOSUB 2240:
    PRINT:PRINT
540 REM manipulador de personajes
550 FOR c=1 TO 6
560 IF FNc(c,10)>0 THEN c$(c,10)=FNm$(c$(c,10),1):
    GOTO 800
570 RESTORE:FOR n=1 TO c×10+c–1:READ
    c$(c,10):NEXT n: REM restablece valor manipulación
    por defecto
580 IF FNc(c,10)=0 THEN GOTO 800
590 GOSUB 2430: GOSUB 5000: REM llama arbol objetos
800 NEXT c
810 GOSUB 4260: IF i$="" GOTO 550
820 GOSUB 2040: GOTO 530
```

Las líneas 2430 y 2440 le asignan a la matriz *c* los diversos valores condicionales que se comprobarán durante el recorrido del árbol. En diversos puntos el manipulador de personajes llama las subrutinas de las líneas 2520, 2620 y 2720

```
2400 REM
2410 REM condiciones
2420 REM
2430 h=FNc(c,8): i=FNc(c,3): j=FNc(c,6): c(1)=ABS(i>
     0): c(2)=ABS((FNb(j,2)=FNc(c,2)) AND (q=1)):
     c(3)=ABS(b$(i,3)="y"): c(4)=ABS
     (FNc(c,3)=FNc(c,6)): c(5)=ABS(b$(i,4)="y")
2440 c(6)=ABS(i=3): c(7)=ABS(FNc(c,5)> 5):
     c(8)=ABS(FNc(c,5)> 2):
     c(9)=ABS(VAL(c$(c,9))=1):
     c(10)=ABS(FNc(x,3)=0):
     c(11)=ABS(FNc(h,2)=FNc(c,2)): c(12)=255
2500 RETURN
2510 REM
2520 REM comprueba escenarios para objetos
2530 REM
2540 f=0:REM establece "bandera hallado" en cero
2550 FOR b=1 TO 12
2560 IF FNb(b,2)=FNc(c,2) THEN f=1:b=12
2570 NEXT b
2580 IF f=1 THEN n=3: GOTO 2600
2590 n=39
2600 RETURN
2610 REM
```



Árbol de manipulación de objetos

Caroline Clayton

Líneas de bifurcación
Vemos aquí el árbol de manipulación de objetos, con los nudos de elección numerados en azul y los nudos terminales en rojo. Los nudos 22 y 23 saltan a subrutinas que determinan el nuevo número de nudo. Cada nudo de elección también muestra (en verde) el número del elemento de la matriz *c* que retiene el valor de la condición a comprobar (ver líneas 2430 y 2440); a todos los nudos aleatorios se les ha asignado el elemento 12. Los nudos terminales están etiquetados con el número de línea de la subrutina a la cual el programa principal le pasa el control después de recorrer el árbol. Próximamente ofreceremos complementos de estos listados para el Spectrum

```
2620 REM comprueba presencia del propietario del objeto:
     si está presente, establece x en el número del
     personaje: vuelve a saltar al árbol
2630 REM
2640 f=0:x=0
2650 FOR m=1 TO 6
2660 IF (FNc(m,2)=FNc(c,2)) AND (FNc(m,6)=FNc(c,3))
     THEN f=1: x=m: GOSUB 2430: m=6
2670 NEXT m
2680 IF f=1 THEN n=15:GOTO 2700
2690 n=39
2700 RETURN
2710 REM
2720 REM selecciona objeto al azar del escenario del
     personaje
2730 REM
2740 b=0
2750 FOR s=1 TO 12
2760 IF FNb(s,2)<>FNc(c,2) THEN GOTO 2780
2770 GOSUB 4180: IF q=1 THEN b=s: s=12
2780 NEXT s
2790 IF b=0THEN GOTO 2750
2800 RETURN
```

Aquí añadimos una rutina de numeros aleatorios para los diversos nudos de elección aleatoria, así como una corta rutina para poner a cero los códigos de c$(n,8) y c$(n,9)

```
4150 REM
4160 REM rutina número aleatorio
4170 REM
4180 q=INT(RND(1)*2)+1: RETURN
4190 REM
4200 REM pone a cero códigos personajes
4210 REM
4220 c$(c,8)="0": c$(c,9)="0": RETURN
4230 REM
4240 REM comprueba si se ha pulsado alguna tecla
4250 REM
4260 i$=INKEY$: RETURN
```

Estas líneas recorren el árbol (5030 a 5060) y seleccionan las diversas rutinas tal como lo determinan los nudos terminales (línea 5090). Observe la línea 5080, que selecciona dos subrutinas que devuelven nuevos valores de nudos y vuelve a saltar al proceso de recorrido del árbol

```
5000 REM rutinas árbol objeto
5010 p=0: REM poner a cero bandera imprimir
5020 IF FNc(c,2)=r THEN p=1
5030 n=1: REM empezar en el nudo 1
5040 IF n>21 GOTO 5070
5050 k=c(k(1,n))+: IF k(1,n)=12 THEN GO SUB 4180:
     k=q
5060 n=t(1,n,k):GOTO 5040
5070 IF n>=24 GOTO 5090
5080 ON (n-21) GOSUB 2540, 2640: GOTO 5040
5090 ON (n-23) GOTO 5100, 5130, 5160, 5180, 5210,
     5240, 5260, 5270, 5280, 5300, 5310, 5330, 5340,
     5360, 5370, 5430
5100 GOSUB 2740: c$(c,3)=STR$(b)
5110 IF p=1 THEN PRINT c$(c,1);" recoge";b$(b,1):
     PRINT
5120 b$(b,2)="0": c$(c,9)="4": RETURN
5130 c$(c,3)=c$(c,6)
5140 IF p=1 THEN PRINT c$(c,1);" recoge";FNi$: PRINT
5150 b$(VAL(c$(c,3)),2)="0"; c$(c,9)="4": RETURN
5160 IF p=1 THEN PRINT c$(c,1);" bebe un sorbo
     de";FNi$: PRINT
5170 c$(c,4)=FNm$(c$(c,4),-1): RETURN
5180 GOSUB 4180:IF (p=1) AND (q=1) THEN PRINT
     c$(c,1);" se está comiendo el bocadillo.":PRINT
5190 c$(c,4)=FNm$(c$(c,4),-2): c$(c,9)="6": GOSUB
     4180:IF q=1 THEN GOSUB 4220
5200 RETURN
5210 IF p=1 THEN PRINT c$(c,1);" prueba vacilando un
     bocado de la empanada, gruñe y la tira al suelo.":
     PRINT
5220 g=c: REM establece bandera comida empanada
5230 c$(c,3)="0": c$(c,4)=FNm$(c$(c,4),10):
     b$(3,2)=c$(c,2): RETURN
5240 IF p=1 THEN PRINT c$(c,1);" suelta";FNi$: PRINT
5250 b$(VAL(c$(c,3)),2)=c$(c,2): c$(c,3)="0":
     RETURN
5260 c$(c,5)=FNm$(c$(c,5),-1): RETURN
5270 GOSUB 5240: RETURN
5280 IF p=1 THEN PRINT c$(c,1);"
     arroja";b$(VAL(c$(c,3)),1);" a "; c$(x,1): PRINT
5290 c$(x,4)=FNm$(c$(x,4),1):b$(VAL(c$(c,3)),2)
     =c$(c,2):c$(x,8)=STR$(c): c$(x,9)="5":
     c$(c,3)="0":RETURN
5300 GOSUB 4220: RETURN
5310 IF p=1 THEN PRINT "Creo que he tomado tu bebida,
     le dice ";c$(c,1);" a ";c$(x,1): PRINT
5320 c$(c,8)=STR$(x): c$(c,9)="2": RETURN
5330 c$(c,4)=FNm$(c$(c,4),2): RETURN
5340 IF p=1 THEN PRINT c$(c,1);" le da"; FNi$;" a
     ";c$(x,1): PRINT
5350 c$(x,3)=c$(c,3): c$(c,3)="0": c$(x,8)=STR$(c):
     c$(x,9)="1": RETURN
5360 GOSUB 4220: RETURN
5370 IF p=0 GOTO 5420
5380 IF p=1 THEN PRINT c$(c,1);" le agradece
     ebriamente a ";c$(VAL (c$(c,8)),1);" que le haya
     devuelto su bebida"
5390 GOSUB 5220: RETURN
5400 RETURN
```

La línea 6230 retiene los datos para el árbol de manipulación de objetos. La línea 220 busca los valores en grupos de tres, asignándolos a cada nudo por turno: el subíndice del elemento de la matriz *c* que retiene la condición a comprobar, el número del nudo al cual debe bifurcar si la condición es verdadera, y el nudo al cual debe saltar si la condición es falsa

```
6200 REM
6210 REM datos árbol objeto
6220 REM
6230 DATA 1, 2, 22, 12, 5, 4, 2, 7, 6, 3, 9, 8, 4, 11, 10,
     12, 39, 24, 12, 6, 25, 5, 12, 39, 6, 13, 27, 12, 23,
     29, 12, 30, 14, 12, 26, 39, 12, 28, 39, 12, 31, 17,
     7, 18, 16, 8, 39, 19, 9, 21, 39, 10, 36, 20, 12, 33,
     32, 12, 35, 34, 11, 38, 37
```

Después del evento

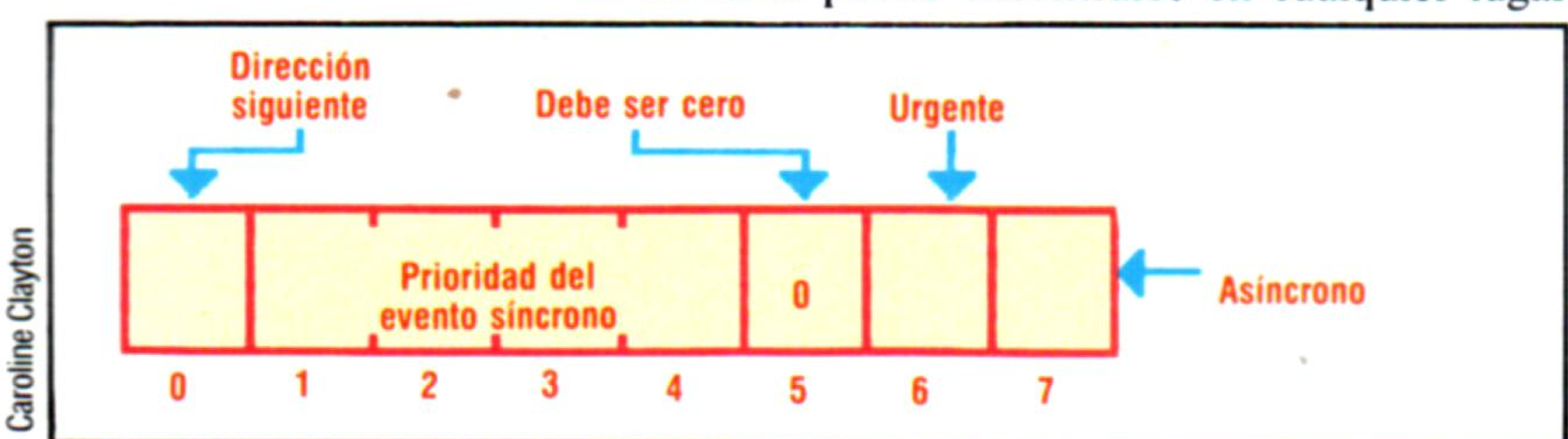
Una vez analizadas las interrupciones y los eventos, ahondaremos en la utilidad que aportan estos últimos al programador en lenguaje máquina

Sincronicidad
Estos dos diagramas de flujo muestran la sucesión de operaciones que realizan el sistema operativo (diagrama superior) y el programa principal (inferior) para procesar los eventos síncronos

Sistema operativo
Evento de lanzamiento
Se lanza el evento
¿Es >0 el cont. eventos?
No
Sí
El evento se coloca en la cola de eventos síncronos
Conclusión

Ya hemos visto cómo el OS del Amstrad accede a los eventos de software a través de un *bloque de eventos*, que consiste en siete bytes contiguos situados en cualquier sitio dentro del bloque central de 32 K de la RAM. Los bloques de eventos son establecidos por el usuario llamando a la rutina KL_INIT_EVENT en $BCEF, siempre que se hayan reservado previamente los siete bytes necesarios. La rutina es llamada contando con que el registro pareja HL contiene la dirección del bloque, B contiene el tipo de evento (en forma de bits significativos, como se muestra más abajo), C contiene la ROM seleccionada (0 si es en RAM) y DE contiene la dirección de la rutina que ha de ser llamada. El listado que proporcionamos es un ejemplo de esta operación.

En el diagrama puede verse la clase de evento tal y como se pasa al registro B. Una vez inicializado un evento, el momento exacto en que es llamada la rutina por el sistema operativo dependerá del tipo (síncrono, asíncrono) y de la prioridad (normal, urgente) del evento. Generalmente la rutina que ha de llamarse puede encontrarse en cualquier lugar



Caroline Clayton

de la RAM o en cualquier ROM y la dirección del campo del usuario en el bloque de eventos se pasa a la rutina del evento para su propio uso.

La manera como maneja el sistema operativo los eventos síncronos y asíncronos es muy distinta, por lo que será mejor que las describamos por separado (aunque ambos tipos, como es obvio, pueden emplearse para cualquier aplicación dada).

Los *puntapiés* o incrementos del contador de eventos pueden venir de una de estas cuatro fuentes distintas: la interrupción rápida de reloj, la interrupción de reloj, la interrupción de retorno de cuadro o la entrada de bloque de saltos KL_EVENT. Las tres interrupciones de temporizador tienen cada una una lista asociada de eventos que necesitan ser lanzados cuando ocurre la interrupción. Las rutinas para establecer las listas son llamadas a través de las entradas del bloque de saltos. Se puede inicializar un nuevo bloque de eventos y añadirlo a una lista, o añadir un bloque preexistente a cualquiera de las listas. Los bloques de eventos pueden ser establecidos e inicializados por separado mediante KL_INIT_EVENT. La rutina KL_EVENT es de uso general y también puede servir para lanzar un evento.



Eventos asíncronos

Asociada con los eventos *normales* encontramos una cola pendiente de eventos de interrupción. Esta cola se emplea para que contenga todos los eventos que han sido lanzados durante una interrupción externa.

Consideremos como ejemplo un bloque de eventos asíncronos normales que ha sido dispuesto para ser lanzado por la interrupción rápida de reloj. Cuando ocurre la interrupción rápida de reloj, el sistema operativo busca los eventos que han de lanzarse. En este caso, el evento será lanzado. Si, tras el lanzamiento, el contador de eventos es mayor que cero, el evento no será tratado inmediatamente, sino que será colocado en la cola de espera de eventos de interrupción. Una vez que el sistema operativo ha realizado todas las tareas relacionadas con la interrupción rápida de reloj, se irán llamando una a una las rutinas que están en la cola de espera de eventos de interrupción. Dado que la interrupción rápida de reloj es reactivada antes de llamar las rutinas de eventos, se anotará todo lanzamiento ulterior. Por ello, la rutina puede emplear el tiempo que sea sin que pierda ninguno de los lanzamientos posteriores. El contador se decrementa una vez llamada la rutina del evento.

Si se ha lanzado un evento asíncrono *urgente* y el contador es mayor que cero, éste no se coloca entonces en la cola pendiente de eventos sino que su rutina de eventos es llamada inmediatamente mientras son desactivadas las interrupciones. Sin embargo, si la rutina de eventos es demasiado larga, entonces se perderá cualquier interrupción externa posterior (por lo tanto, la rutina ha de ser tan corta como se pueda). Por lo general, este tipo de evento no se utiliza.

Eventos síncronos

El programa principal decide cuándo han de procesarse los eventos síncronos. Por esto, el sistema operativo lanza sencillamente el evento y, si es mayor que cero, lo pone en la cola pendiente de eventos síncronos según la prioridad que se le asigne en el bloque de eventos. Los eventos síncronos urgentes se sitúan antes que los de tipo normal.

El sistema operativo proporciona varias entradas del bloque de saltos al programa principal para procesar los eventos síncronos, para limpiar la cola si es el caso y para impedir que ocurran eventos particulares. Además, si es necesario, se pueden desactivar todos los eventos síncronos normales.

Cuando el programa principal decide procesar los eventos síncronos, éste efectúa las tareas representadas en los diagramas de flujo. Primeramente se llama la entrada del bloque de saltos KL_NEXT_SYNC para obtener el siguiente evento de turno en



la cola. El evento es entonces procesado mediante la llamada a KL_DO_SYNC; esta rutina busca la dirección de la rutina de eventos en el bloque de eventos y la llama. Entonces se llama la entrada KL_DONE_SYNC para indicar al sistema operativo que ha concluido el procesamiento correspondiente a ese evento. En este punto es decrementado el contador de eventos, y si éste sigue siendo mayor que cero, el bloque de eventos vuelve a ser colocado en la cola de eventos síncronos.

Desactivación de eventos

En un determinado momento puede que sea necesario desactivar o evitar que ocurran determinados eventos particulares. El sistema operativo permite esto con numerosas entradas del bloque de saltos. Para eventos asíncronos, la entrada KL_DISARM_EVENT pone a un valor negativo el contador de eventos relativo a un bloque de eventos dado, y por lo tanto impide que cualquier lanzamiento ulterior llame a la rutina de eventos.

Tres son las entradas que permiten desactivar los eventos asíncronos.

La primera entrada, KL_SYNC_RESET, limpia todos los eventos pendientes de la cola de espera de eventos síncronos pero no altera el contador de eventos. Esto desactiva efectivamente el evento, dado que el contador de eventos no puede decrementarse a menos que el evento esté en la cola. Otra entrada, KL_DEL_SYNCHRONOUS, desactiva y elimina todo evento que pueda encontrarse en la cola.

La última entrada, KL_EVENT_DISABLE, detiene toda formulación de los eventos síncronos normales que se haya colocado en la cola, pero todavía permite que se lancen los eventos.

Los eventos pueden parecer complicados al principio, pero protegen al programador de las complicaciones que normalmente acompañan las interrupciones, aunque se ha de tener mucho cuidado cuando se emplean los eventos asíncronos urgentes para que no interfieran los datos con el programa principal.

Zumbador de fondo

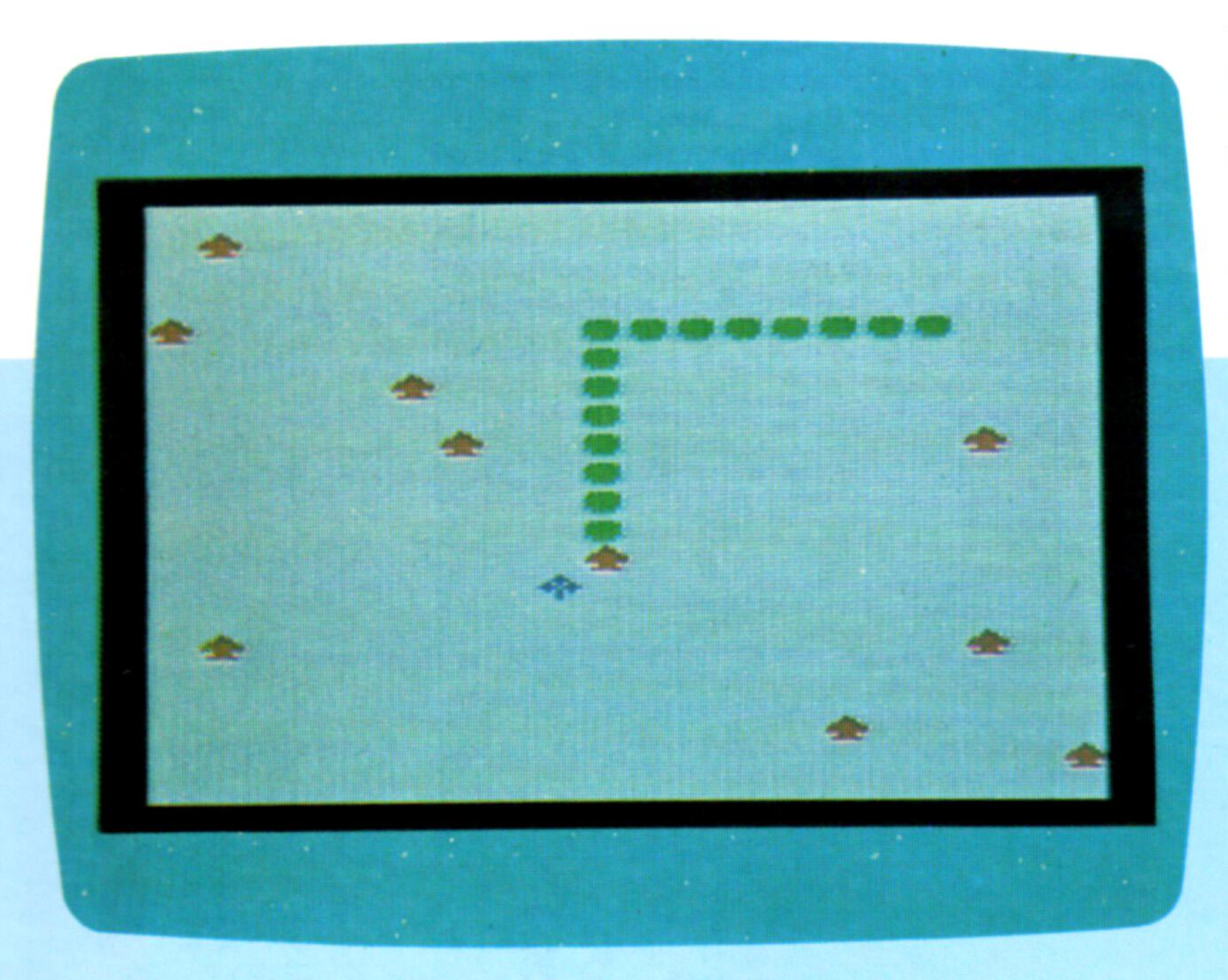
Este listado es un ejemplo de cómo se emplean los eventos, utilizando las técnicas explicadas en el texto. Se establece un evento de reloj y se utiliza como temporizador, lo que hace que un evento cuente los segundos. Éste a su vez establece un evento que obliga al ordenador a emitir un zumbido una vez por segundo. El programa ha de llamarse a su dirección assembly para inicializar los eventos.

Obsérvese que el zumbido continúa de fondo cuando se está ejecutando un programa en BASIC (aunque puede que sucedan extraños efectos si el mismo programa en BASIC emplea a su vez el generador de sonido)

```
;Este programa demuestra el empleo de varios
;tipos de eventos para generar un reloj sencillo que
;emite un sonido cada segundo
;
INIT.EVENT:          EQU £BCEF ; KL_INIT_EVENT
ADD.TICK:            EQU £BCE9 ; KL_ADD_TICKER
KICK:                EQU £BCF2 ; KL_EVENT
TXT.OUT              EQU £BB5A ; TXT_OUTPUT
;
BLEEP:               EQU £07
;
;Primero se inicializan los bloques de eventos
;
                     ORG £8000
;
                     LD   HL, TICK    ;inicio con reloj
                     LD   B, £81      ;asincr, urgente
                     LD   C, 0        ;sin selecc. rom
                     LD   DE, FRAC    ;direccion de la rutina
;
                     CALL INIT.EVENT
;
;Se conserva BC
;
                     LD   HL, SEC.    ;contador segundos
                     LD   DE, SECND   ;direccion rutina
                     CALL INIT.EVENT
;
                     LD   HL, BP      ;rutina sonido
                     LD   DE, BEEP    ;direccion rutina
                     CALL INIT.EVENT
;
;Se cubre ahora la Rutina Reloj
;
                     LD   HL, FRBLK   ;direccion bloque reloj
                     LD   DE, 1       ;contador
                     LD   BC, 1       ;recarga
                     CALL ADD.TICK    ;lo registra
;
                     RET
;
;Rutinas procesamiento eventos
;
FRAC:
;mantiene un contador de 1/50 -avo de segundo
;
                     LD   HL, SEC50   ;apunta al contador
                     LD   B, 50       ;un segundo?
                     CALL TEST
                     RET  NZ          ;si no es asi, concluir
                     LD   HL, SEC     ;lanza siguiente segundo
                     CALL KICK
                     RET
;
SECND:
;Es llamada una vez por segundo - lanza un zumbido
                     LD   HL, BP      ;lanza evento zumbido
                     CALL KICK
                     RET
;
BEEP:
;hace zumbido
                     LD   A, BLEEP    ;envia un caracter sonido
                     CALL TXT.OUT
                     RET
;
TEST:
;comprueba contador
;B contiene valor necesario en entrada
;HL apunta a posicion almacenamiento contador
;
                     INC  (HL)        ;lo actualiza
                     LP   A, (HL)     ;lee contador
                     CP   B           ;conclusion
;
                     RET  NZ          ;no, retorno
                     XOR  A           ;si verdadero pone cero
                     LD   (HL), A     ;restablece contador
                     RET              ;y retrocede
;
;ahora los bloques de eventos y reloj
;
FRBLK:               DEFS 6           ;espacio bloque reloj
;
TICK:                DEFS 7           ;bloque evento reloj
;
SEC:                 DEFS 7           ;bloque evento 1 seg
;
BP:                  DEFS 7           ;bloque evento zumbido
;
SEC50:               DEFB 0           ;contador 1/50 seg
```

Ciempiés

Esta versión de este conocido juego ha sido escrita en BASIC para el microordenador Commodore Vic 20



Trate de dirigir su ciempiés robot durante el mayor tiempo posible. Él ha de alimentarse con las flores azules (las flores rojas están envenenadas) sin salirse nunca del cuadro ni recortar su propio cuerpo. La dificultad reside en que su longitud aumenta una unidad después de cada comida, lo cual hace que los desplazamientos sean cada vez más comprometidos.

```
10 REM *************
20 REM * CIEMPIES *
30 REM *************
50 GOSUB 1000
100 GET X$
110 D1=(X$=“A”)-(X$=“S”)+22*((X$=“W”)-(X$=“Z”))
120 IF D1<>0 THEN D0=D1
125 IF FL=1 THEN FL=0:GOTO 170
130 POKE A(0),CN
140 FOR I=0 TO L
150 A(I)=A(I+1)
160 NEXT I
170 A(L)=A(L-1)+D0
180 C=PEEK(A(L))
190 IF C=CB THEN 300
200 IF C<>32 AND C<>42THEN 500
210 POKE A(L),MP
220 POKE A(L)+M,MC
230 GOTO 100
300 GOSUB 2000
320 POKE A(L),MP
330 POKE A(L)+M,MC
340 L=L+1
350 FL=1
360 GOTO 100
500 PRINT:PRINT:PRINT
505 GOSUB 700
510 PRINT TAB(5)“PUNTUACION:”L*10-70
520 PRINT:PRINT:PRINT
530 PRINT TAB(5)“ OTRA ?”
540 GET X$
550 IF X$<>“” THEN 540
560 GET X$
570 IF X$=“” THEN 560
580 IF X$<>“N” THEN RUN
590 PRINT CHR$(147);
600 END
700 FOR I=1 TO 6
710 POKE A(L),CN
730 FOR J=1 TO 200
740 NEXT J
750 POKE A(L),42
760 POKE A(L)+M,4
770 FOR J=1 TO 200
780 NEXT J
790 NEXT I
800 RETURN
1000 PRINT CHR$(147);
1010 GOSUB 1200
1020 GOSUB 1400
1030 GOSUB 1600
1040 GOSUB 1800
1050 GOSUB 2000
1190 RETURN
1200 CN=32
1210 CH=65
1220 HC=2
1230 BC=6
1240 DIM A(70)
1250 M=30720
1260 MP=81
1270 MC=5
1280 L=7
1290 D0=1
1300 CB=88
1390 RETURN
1400 FOR I=0 TO L
1410 A(I)=7923+I
1420 POKE A(I),MP
1430 POKE A(I)+M,MC
1440 NEXT
1450 RETURN
1600 FOR I=0 TO 21
1610 POKE 7680+I,160
1620 POKE 7680+I+M,0
1630 POKE 8164+I,160
1640 POKE 8164+I+M,0
1650 NEXT
1660 FOR I=1 TO 22
1670 POKE 7680+I*22,160
1680 POKE 7680+I*22+M,0
1690 POKE 7701+I*22,160
1700 POKE 7701+I*22+M,0
1705 NEXT
1710 FOR I=1 TO 10
1720 GOSUB 1800
1730 POKE P,CH
1740 POKE P+M,HC
1750 NEXT
1760 RETURN
1800 P=INT(RND(TI)*440)+7702
1810 IF PEEK(P)<>32 THEN 1800
1820 RETURN
2000 GOSUB 1800
2010 POKE P,CB
2020 POKE P+M,BC
2030 RETURN
```

Editorial Delta, S.A.



10104
9 788485 822836